1-1-2014

Exploring Linguistic Constraints in Nlp Applications

Qiuye Zhao
University of Pennsylvania, qiuyezhao@gmail.com

Exploring Linguistic Constraints in Nlp Applications

Abstract
The key argument of this dissertation is that the success of an Natural Language Processing (NLP) application depends on a proper representation of the corresponding linguistic problem. This theme is raised in the context that the recent progress made in our field is widely credited to the effective use of strong engineering techniques. However, the intriguing power of highly lexicalized models shown in many NLP applications is not only an achievement by the development in machine learning, but also impossible without the extensive hand-annotated data resources made available,

which are originally built with very deep linguistic considerations.

More specifically, we explore three linguistic aspects in this dissertation: the distinction between closed-class vs. open-class words, long-tail distributions in vocabulary study

and determinism in language models. The first two aspects are studied in unsupervised tasks, unsupervised part-of-speech (POS) tagging and morphology learning, and the last one is studied in supervised tasks, English POS tagging and Chinese word segmentation. Each linguistic aspect under study manifests

itself in a (different) way to help improve performance or efficiency in some NLP application.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Computer and Information Science

First Advisor
Mitchell P. Marcus

Keywords
Chinese word segmentation, closed-class words, long-tail distribution, Morphology learning, natural language processing, Unsupervised POS tagging

Subject Categories
Computer Sciences

EXPLORING LINGUISTIC ASPECTS IN NLP APPLICATIONS

Qiuye Zhao

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2014

Supervisor of Dissertation

_____

Mitchell P. Marcus
Professor of Computer and Information Science

Graduate Group Chairperson

_____

Lyle Ungar
Professor of Computer and Information Science

Dissertation Committee:
Aravind Joshi, Professor of Computer and Information Science
Chris Callison-Burch, Assistant professor of Computer & Info. Science
Charles Yang, Associate professor of linguistics
Dekai Wu, Professor of Computer Science and Engineering, HKUST

Acknowledgements

Given the level of freedom by my father, Dayu Zhao, I kind of feel obligated to live a life distinguished from one that ”has to” succeed. Given the level of freedom by my advisor, Mitch Marcus, I kind of feel obligated to work on something distinguished from those that ”have to” be good. Given the birth of my son, who is 8 months older than this, I guess this dissertation will be the last piece of my work that doesn't ”have to” be good. Special thanks to Charles Yang, my linguistic professor and perhaps the only other one who takes this dissertation as a linguistic work. However, the supervisor of my only formal linguistic work is Charles' wife, Julie Legate, so thanks to their whole family. One great thing about the NLP group at Penn is that you are not alone, and Professor Aravind Joshi is the one that surprises me with so insightful comments on students' work. Even though I didn't choose to stay in the U.S. , this is a great place with very nice people living on the (geographically) isolated but (in spirit) open land. Most of the friends I make here will stay, whose names I know I will forget in years, but the good time shared is what mattered.

ABSTRACT

EXPLORING LINGUISTIC ASPECTS IN NLP APPLICATIONS

Qiuye Zhao

Mitchell P. Marcus

The key argument of this dissertation is that the success of an Natural Language Processing (NLP) application depends on a proper representation of the corresponding linguistic problem. This theme is raised in the context that the recent progress made in our field is widely credited to the effective use of strong engineering techniques. However, the intriguing power of highly lexicalized models shown in many NLP applications is not only an achievement by the development in machine learning, but also impossible without the extensive hand-annotated data resources made available, which are originally built with very deep linguistic considerations. More specifically, we explore three linguistic aspects in this dissertation: the distinction between closed-class vs. open-class words, long-tail distributions in vocabulary study and determinism in language models. The first two aspects are studied in unsupervised tasks, unsupervised part-of-speech (POS)

tagging and morphology learning, and the last one is studied in supervised tasks, English POS tagging and Chinese word segmentation. Each linguistic aspect under study manifests itself in a (different) way to help improve performance or efficiency in some NLP application.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The key argument of this dissertation is that the success of a Natural Language Processing (NLP) application depends on a proper representation of the corresponding linguistic problem. This theme is raised in the context that the recent progress made in our field is widely credited to the effective use of strong engineering techniques. However, the intriguing power of highly lexicalized models shown in many NLP applications, is not only an achievement by the development in machine learning, but also impossible without the extensive hand-annotated data resources available. In the original work on building data resources, e.g. (Marcus et al., 1993; Miltsakaki et al., 2004), deep concerns in linguistic aspects are well addressed. Then the representations used in these data resources somehow standardize the formulation of many NLP problems. Therefore, follow-up work is more concentrated on the engineering aspects than the linguistic aspects of NLP problems.

Besides my own curiosity in language (and taking my engineering work as an approach to explore linguistic phenomena), this dissertation is devoted to the study of several funda-

mental linguistic aspects in NLP applications for at least two practical reasons:

1. **Unsupervised learning.** Acquiring linguistic structures from raw data with little or minimal supervision should never be a less important task than its supervised counterpart that takes sufficient training examples as granted. The study of unsupervised learning is not only for a cognitive interest or because of under-resourced languages, but also for complementing supervised learning. As in (semi-)supervised tasks, we may use unsupervised learning techniques to acquire a different form of representation than provided or to make use of raw materials in addition to limited resources. As shown by our study, when data resource is limited, the success of learning is more sensitive to a proper linguistic understanding of the corresponding problem.

2. **Efficiency.** Even though a trade-off between accuracy and efficiency is commonly accepted, we never sacrifice efficiency for accuracy, or vice versa! So as to achieve this goal, for various tasks, we propose to decompose the problem in a linguistically sensible way, thus constraining the searching space but avoiding the pruning of good hypotheses. This advantage in efficiency suggests that the application of strong engineering techniques does not prevent us from getting a cognitive sense of the corresponding study of language. Moreover, our contribution to efficiency is based on our study of fundamental linguistic constraints, thus is in addition to the traditional techniques for seeding-up searching.

More specifically, we explore three linguistic aspects: the distinction between closed- vs. open-class words, long-tail distributions in vocabulary study and determinism in lan-

guage models. The first two aspects are studied in two unsupervised tasks respectively and the last one is studied in supervised tasks.

Functional elements and unsupervised POS tagging.

Functional words are usually considered as closed-class words in the sense that they can be enumerated in a short list. When there is no such a list provided, functional words are usually approximated by the most frequent words. These two views are taken as granted in our field without further consideration, since functional words do not play any particularly interesting role in highly lexicalized models. From a syntactic point of view, functional words occur more often at the edge of elementary linguistic structures, thus observed in more diverse contexts due to the compositionality of language. We propose to distinguish functional words by their distinctively high contextual diversity, and propose a bootstrapping algorithm acquiring functional words from raw text only (Zhao and Marcus, 2011). This bootstrapping algorithm can also be applied to morphology learning, thus able to acquire functional elements in both free and bound morphemes.

Furthermore, we propose a feature-based analysis of part-of-speech (POS) tags, then the tagging problem is not necessarily formulated as a sequence labeling problem. Instead, we argue that, if we only concern with coarse lexical tags, except for one binary syntactic feature that needs to be disambiguated during tagging, all other features of POS tags, such as semantic and morphological features, can be settled by global learning processes (as opposed to locally disambiguation). For example, semantic features can be set by POS induction and morphological features can be set by morphology learning. The distinction

between closed- and open-class words is then crucial in practicing such a feature-based view for unsupervised POS tagging. First of all, the decomposing story is only for lexical categories, i.e. open-class words, in contrast, closed-class words are much more ad-hoc but enumerable. Second, in a distributional clustering for POS induction, we only use functional elements in the local context of a word to represent it in the feature space. Third, so as to make a better use of unambiguous cases to establish disambiguation rules, the tagging system is designed to learn and tag open-class words first and dealing with closed-class words afterwards. Finally, the disambiguation rules for open-class words are conditioned on closed-class words in their local contexts.

Without any lexicon input, the totally unsupervised POS tagging system tags 6 lexical categories with a rather promising performance, and with the input of a closed-class lexicon, which contains only about 0.6% word types of the full lexicon, the proposed two-stage unsupervised POS tagging system can achieve a tagging accuracy comparable to the so-called unsupervised models that requires the input of full lexicons. Moreover, since we first highlighted the distinction between closed- and open-class words in (Zhao and Marcus, 2009), the idea of distinguishing closed-class words from other words has achieved more and more attention in following works on unsupervised POS tagging by others, such as (Graca et al., 2009; Teichert and Daume, 2010; Moon et al., 2010) etc.

Long-tail distribution.

The algorithm we propose for acquiring functional words can be generalized to acquire either functional words or morphological endings in English, with different definitions of

'contexts'. This bootstrapping algorithm is motivated by Chan (2008)'s work on morphology learning , which utilizes the long-tail patterns observed in word distribution as well as in morphology. In previous study of vocabulary or morphology, such a long-tail distribution is usually considered as power-law, also known as Zipf's law (Zipf, 1949).

So as to deal with real text input, which reflects long-tail word distribution, we have been open to rule-based methods, maximum likelihood methods and Bayesian methods. And only the last proposed Bayesian model with log-normal assumptions handles token-based input well (Zhao and Marcus, 2012b). Even though previously proposed Bayesian models that generate power-law distributions can also transforming away word frequencies, e.g. (Goldwater et al., 2006; Chahuneau et al., 2013), our proposed model is the first one that actually takes advantage of word frequencies for a toke-based evaluation and performs better with real text input than with type-based input. Since word distribution is conventionally studied by power-law distributions, we devote a section to examine whether there is any theoretical aspect favoring Zipf's law over log-normal distributions in vocabulary study, and discover none.

More specifically, for learning inflections on English verbs, we first try a rule-based approach, which utilizes the acquired morphological transformations by the proposed bootstrapping algorithm. This rule-based approach learns from type-based input only, i.e. the input of distinct word types. Then we try both the Expectation Maximization (EM) learning and Gibbs sampling for the estimation of our morphology models. When multinomial distributions are assumed for the sake of simplicity in computation, both inferences succeed with type-based input only, but not handling real text input that reflects the long-tail

word distribution. Thus we propose to assume log-normal distributions for morpheme and word frequency, and run Gibbs sampling for inference.

Deterministic constraints.

From an engineering point of view, searching the hypothesis space in a 'deterministic' way is no more than a special case of pruning. From a linguistic point of view, the syntactic structures of language are deterministic in a sense of being universally hard-coded, and nondeterminism of language only lies in the ambiguity of lexical items. In highly lexicalized statistical models, this understanding is hard to be implemented and in non-lexicalized models, the stat-of-art performance is hard to be achieved. Therefore, our interest in the determinism of language leads to a study of deterministic constraints.

More specifically, we propose to learn deterministic constraints by data-driven training and use them to constrain probabilistic inference. When the deterministic constraints are learnt from the same representation as the probabilistic model, this idea appears non-distinguishable with pruning. However, when the deterministic constraints are learnt from a different representation of the problem, they may be used to constrain probabilistic inferences that are not suitable for direct pruning.

Whether deterministic constraints can be learnt are very sensitive to the way they are represented; and if the hypothesis space of one representation is hard to be pruned directly, we need to explore other representations of the same problem for learning deterministic constraints. For example, for the problem of Chinese word segmentation, we propose to reconsider the word-based model, which draws much less attention than the character-

based model in recent works. We propose an Integer Linear Problem (ILP) formulation for the word-based model of the word segmentation problem, and constrain the inference of valid segmentations by character-based constraints. This model (Zhao and Marcus, 2012a) achieves the state-of-art performance for Chinese word segmentation, and by applying the deterministic constraints, the ILP solver is speeded-up by 107 times. We have also applied the same idea to supervised POS tagging (Zhao and Marcus, 2012a). Even for a searching with beam-size 5, which is already much faster than a full searching, the overall tagging can still be speeded-up by another 10 times, by applying the deterministic constraints.

We are going to explore the distinction between closed- and open-class words for the unsupervised POS tagging problem in Chapter 2, explore long-tail distributions for morphology learning in Chapter 3, and explore deterministic constraints in Chapter 4 on English POS tagging and Chinese word segmentation.

Chapter 2

Functional elements & Unsupervised POS tagging

2.1 Introduction

In this chapter [1] , we are going to describe a bootstrapping algorithm acquiring functional elements from raw text input only. There are two forms of functional elements considered:

- closed-class words (vs. open-class words), and
- morphological transformations, e.g. inflectional endings in English.

We are not aware of any previous work exploring the acquisition of closed-class words; instead, closed-class words are usually considered as the most frequent words in text. In

[1] This chapter extends (Zhao and Marcus, 2011), which are re-organized to Section 2.4,2.5 and 2.8, and (Zhao and Marcus, 2009), which is basically Section 2.7.

this work, we show that, as compared to the most frequent words, the set of words acquired by our proposed algorithm makes much more sense as closed-class words. More important, the acquired set of words can serve as input to more advanced acquisition tasks, and this idea is shown to work with our experiments on unsupervised POS tagging. We propose to build an unsupervised POS tagging system based on the distinction between closed- vs. open-class words. With the experiments on this unsupervised system, we show that

1. given a minimized dictionary containing closed-class words only, we can build a POS tagging system comparable to the state-of-art unsupervised taggers that require the input of a full dictionary;
2. and when pipelined with the acquisition of functional elements, we can build a totally unsupervised system that achieves a POS tagging accuracy above 85% for open-class words, requiring no other input than raw text.

Note that, for historical reasons, when we talk about 'unsupervised' POS tagging, the use of word 'unsupervised' emphasizes the lack of token-based annotation data in the contrast to supervised learning. Thus, in earlier literature on unsupervised POS tagging, e.g. (Smith and Eisner, 2005) and (Goldwater and Griffiths, 2007), a dictionary containing possible POS tags for each word is usually assumed to be provided. However, if the available resource is limited to raw text only, we consider the learning as 'totally unsupervised' in our context. Totally unsupervised POS tagging attracts more and more attention recently, e.g. (Abend et al., 2010), (Reichart et al., 2010), (Moon et al., 2010) etc.

The acquisition of morphological transformations is better known as morphological learning. There is a wealth of literature on it, e.g. (Chan, 2008), (Lignos, 2010) etc., however, we are the first to consider the homogeneous relationship between morphological learning and the acquisition of closed-class words. When instantiated with a proper definition of 'contextual diversity', the proposed algorithm is able to acquire closed-class words or morphological transformations correspondingly, thus exhibiting the relationship between the free and bound forms of functional elements. The acquisition output of morphological transformations may also be used in experiments on unsupervised POS tagging.

First, we give a rough overview on the distinction between closed- and open-class words in Section 2.3. In Section 2.4, we propose a bootstrapping algorithm acquiring closed-class words from raw text only. In Section 2.5, we show how the proposed bootstrapping algorithm is applied to morphological learning. In Section 2.7, we propose an unsupervised POS tagging system based on the distinction between closed- vs. open-class words. Finally, in Section 2.8, we integrate the acquisition of functional elements and unsupervised POS tagging, delivering a totally unsupervised POS tagging system that requires raw text only.

2.2 Related work

Acquisition of the closed-class lexicon is not a widely-studied research topic, since the distinction between closed- and open-class words is not that interesting in lexicalized models, which are the mainstream models for supervised learning. However, we argue that this categorical distinction deserves attention in unsupervised learning systems, and propose

a state-of-art unsupervised POS tagging system to support our argument. We immediately have followers that incorporate this distinction in their models for unsupervised POS tagging. For example, two distributional properties of closed-class words are captured by Crouching Dirichlet, Hidden Markov Model (CDHMM) in (Moon et al., 2010): higher variance across contexts (e.g. documents) and more peaked emission properties. A state alphabet, which is mapped to POS tags later for evaluation, is a union of disjoint content (open-class) states and function (closed-class) states, and in such a composite model, the priors for the emission and transition at each step depend on the category of the generated state. Similarly, in (Teichert and Daume, 2010), the generative process is also modified to incorporate a binary variable for openness, and if a word is sampled from the closed-class, the smoothing parameter is smaller. In these works, the binary categorical distinction is estimated as part of the generative model, thus there is no need for building a closed-class lexicon. Our experiments on acquisition of the closed-class lexicon was compared to the token-based evaluated result in (Graça, 2011), which uses this task for exploring their learning framework on mapping annotations of other languages to the target language.

Furthermore, we propose to evaluate acquired closed-class lexicons in the task of unsupervised POS tagging. Our unsupervised POS tagging system consists of two parts: POS induction on lexical categories and the learning of a POS disambiguation model without labeled data. There are quite a few previous works on the learning of POS disambiguation models from unlabeled data, but taking a complete or partial dictionary as input. The state-of-art performance in this line of work is achieved by Ravi and Knight (2009), which constrains the EM learning of a HMM-based tag model with the smallest grammar acquired by

solving an Integer Programming (IP) problem whose objective function is explicitly coded to minimize the grammar size. There are also other efforts on using the EM algorithm for training HMM-based tag models when labeled data are not available, for example, Goldberg et al. (2008) intervene with expert knowledge to set up good initial conditions for EM learning, Bayesian framework is also very popular in use so that high-level beliefs about the tag model can be incorporated. For example, in (Goldwater and Griffiths, 2007; Toutanova and Johnson, 2008) Dirichlet priors favoring sparse multinomial distributions are used. Except for the generative tagging models, Smith and Eisner (2005) proposes contrastive estimation for training discriminative tagging models from unlabeled data.

A well-known limitation of above systems is that their learning heavily relies on the input of a dictionary specifying possible tags for each word. When the input dictionary is complete, the unsupervised system proposed in (Ravi and Knight, 2009) even achieves a tagging accuracy comparable to supervised models. However, when the input dictionary is reduced to contain words with a count above a given threshold, tagging performance goes quickly down as the size of the dictionary goes down. Toutanova and Johnson (2008) dealt with incomplete dictionary input by explicitly include a variable of ambiguity class in their generative model and achieved the best performance with incomplete dictionary before us (Zhao and Marcus, 2009). We propose to reduce the input of a dictionary as minimized as a closed-class lexicon containing less than 300 hundred words, and outperforms the best can be achieved with a partial dictionary of more than 2000 words in previous work.

Other than complete or partial dictionaries, other kinds of information are also considered in unsupervised POS tagging. For example, Haghighi and Klein (2006) provide seed

words for each gold POS tag in their prototype-driven learning; and Garrette and Baldridge (2013) constrain the learning by annotations produced in a rush. Moreover, Reichart et al. (2010) add a Zipfian constraint to the class n-gram model described in (Clark, 2003), and we also pay special attention to Zipf's law in word distribution and will discuss it for morphology learning in Chapter 3.

Finally, we plug the acquired closed-class lexicons into our unsupervised POS tagging system and obtain a 'totally' unsupervised tagging system that requires no input of dictionary but raw text data only. The so-called 'totally' unsupervised tagging task is also explored as the problem of POS induction. We also have a POS induction model in our system for building a open-class dictionary, before the learning of POS disambiguation rules. Following a classic work (Schütze, 1993), we simply employ a basic k-means method for clustering lexical categories, since our distributional representation of words is rather compact. Distributional clustering is recently re-considered in (Lamar et al., 2010) and achieves the state-of-art totally unsupervised tagging performance with a one-to-one mapping from induced clusters to gold POS tags. Unlike our representation of words (to be clustered) that depend on distributional statistics of each word only, Lamar et al. (2010) propose latent descriptors of words that also depend on clustering assignment. HMM-based tag models with Bayesian inference are also very popular in this line of work. Besides aforementioned works (Teichert and Daume, 2010; Moon et al., 2010; Toutanova and Johnson, 2008; Goldwater and Griffiths, 2007), Blunsom and Cohn (2011) proposes Pitman-Yor processes for smoothing priors and achieves the state-of-art totally unsupervised tagging performance with many-to-one mapping from induced states to gold POS tags. Sparsity can also be

achieved by posterior regularization as shown in (Graca et al., 2009).

As one may have noticed, it is very confusing to compare works on POS induction and works titled as unsupervised POS tagging. The confusion is caused by the use of tagging-based evaluation measures in literature. As we have mentioned, we can map induced clusters/states to gold POS tags and evaluate the tagged sequences created by the cluster identifiers. There are two mapping strategies popular in use, many-to-one that allows more than one clusters mapped to the same gold tag and one-to-one that allows at most one cluster mapped to each tag. So as to map each induced cluster to the gold tag that is preferred by most of its words, the mappings are actually taking advantage of more information than these unsupervised systems claimed, and that is why we propose to induce clusters corresponding to disjoint lexical categories only. Since most previous work on POS induction use finer tags than we do here, we may appear to be reporting on a simpler task. However, it has proven difficult for these systems to use further agglomerative processing to induce simple distinct syntactic categories which map to POS tags naturally (Christodoulopoulos et al., 2010). Therefore, achieving high accuracy with a smaller tag set is the harder, not easier, task for those systems. On the other hand mapping our output of 6 main lexical categories to an artificial POS tagset requires detailed and ad-hoc supervision beyond what we consider 'natural' assumptions. We leave it to future work to explore which form of output is favorable by more advanced unsupervised tasks.

2.3 Closed-class vs. open-class words

Closed-class words are named so because languages rarely add new items to this vocabulary. They are considered as the bound forms of functional elements, because closed-class words serve to structure a sentence syntactically. In the contrast to open-class words, closed-class words do not have referential meanings that refer to objects or notions. Instead, closed-class words carry grammatical functions and hold the content words together. For English, we consider words of four lexical categories as open-class words: nouns, verbs, adjectives and adverbs. All other categories of words are considered as closed-class words, such as determiners, prepositions, pronouns, conjunctions, auxiliary verbs, and particles.

Besides the closed nature and grammatical roles, closed-class words also have some special phonological characteristics. For example, in English, a lexical word cannot consist of a light syllable alone, but there are closed-class words that do not obey this minimal word constraint, e.g. *I*, *the*, *a*, etc... This economical representation may correlate with the fact that these closed-class words are highly frequently used in all occurrences, and a few types of closed-class words account for a large portion of the word count. For example, with the help of the gold annotation of Part-of-speech tags, we collected 288 closed-class words from the WSJ Penn Treebank. This set is rather small compared to the open-class set containing more than 40000 words; but it accounts for about one-third of the word count in the WSJ corpus. Observing the closed nature and the distinctively high token/type ratio of closed-class words, we propose to consider a set of closed-class words as a minimally required input to unsupervised learning tasks, and explore this idea with experiments on

unsupervised POS Tagging in Section 2.7.

We are not the first interested in making use of the distinction between closed- and open-class words for NLP tasks. For example, it is argued that closed-class words "*carry an array of psychological meanings and set the tone for social interactions*" (Chung and Pennebaker, 2007); therefore, closed-class word frequencies are often used as features in authorship attribution. In code-switching models, it is proposed to put special constraints on the switchability of closed-class items, observing that a bilingual speaker may switch her language for open-class words but not for closed-class words (Joshi, 1982). Moreover, for information retrieval, closed-class words are usually contained in a list of "stop words", so that to be excluded as index terms in the semantic representation of documents.

Perhaps, since the stop-word list can also be built with high-frequency words (Luhn, 1958), high-frequency words are considered as a convenient approximation of closed-class words in NLP applications. However, as we are going to see in Section 2.4 and Section 2.8, for language acquisition tasks, this approximation doesn't lead to the best result. On the other hand, we also have theoretical reasons to distinguish the concept of closed-class words from the observation of high frequency. There is an extensive cognitive literature on the distinction between closed- and open-class words. Their experimental methods include the traditional behavioral study (Bradley, 1978; Gordon and Caramazza, 1982; Biassou et al., 1997), as well as high-tech ways to measure brain responses such as event-related potentials (ERP) (Friederici et al., 1993; Neville et al., 1992), functional magnetic resonance imaging (fMRI) (Friederici et al., 2003) and magnetoencephalography(MEG) (Wang et al., 2008). Both normal subjects and aphasics are studied, and especially, it is the interesting

data on Broca's aphasia that invoke the original study of this topic (Friederici and Schoenle, 1980; Bates and Wulfeck, 1989). It receives so broadly research interests in cognitive science, because this categorical distinction between closed- and open-class words raises issues that are independent of those raised by word frequency effects. Word frequency may be the most studied factor in lexical access and production, however, the aforementioned studies show that word frequency effect does not explain the whole story without considering the distinction between closed- and open-class words. Similar interests can also be found in the study of child language acquisition. Regardless of the high-frequency uses of closed-class words in mother language, the child doesn't use closed-class words until they master a base vocabulary of hundreds of nouns and verbs (Goodman et al., 2008).

As discussed above, the distinction between closed- and open-class words is of both theoretical and practical interests. Thus, it is to our surprise that we are not aware any previous work that take the acquisition of closed-class words as a serious problem. In the next section, we are going to describe a bootstrapping algorithm acquiring closed-class words from raw text only, and in later sections, we will show that this result can be used in further acquisition tasks.

2.4 The acquisition of closed-class words

In this section, we are going to acquire a set of words that distinguish themselves by the diverse types of contexts they can occur in. Inclined to occur in highly diverse contexts is a well-known distributional property of functional elements. For example, determiner

'the' in English may be followed by almost any noun or adjective in text, and inflectional ending *'-ed'* can be concatenated to most verbs to derive past forms. In contrast, content words tent to occur in much more limited contexts. For example, noun 'Fannie' occurs 52 times in the WSJ corpus but only in one proper noun phrase "Fannie Mae". It is not a surprise to see closed-class words occur in more diverse contexts, since they provide structural bones to compose expressions. For example, verb 'compared' is almost always followed by prepositions 'to' or 'with', but the prepositions themselves may be followed by various word types.

2.4.1 The measurement of contextual diversity

We have argued that, as compared to high frequency, high contextual diversity is the property that distinguishes closed-class words better. On the other hand, we have implicitly considered the types of words that ever follow a word w in text as the contextual diversity of the word w. According to this measurement, as measured in the WSJ corpus, the contextual diversity of 'Fannie' is 1 and the contextual diversity of 'compared' is 2. More formally, given a corpus $\mathbb{C}$ of sequences of tokens, for each word type w, we compute its type-based contextual diversity according to the following words as follows:

$$typeC_{following}(w, \mathbb{C}) = \sum_{w' \in \mathbb{C}} \text{\# occur. of } w' \text{ following } w > 0$$

i.e. with function $typeC_{following}$, we count all the word types that are ever seen following a given word type. Even if we only consider contextual relationships between two adjacent words, there is still another alternative to measure the type-based contextual diversity of a

word: we can also count the types of words that are ever seen preceding a word, i.e.

$$typeC_{preceding}(w, \mathbb{C}) = \sum_{w' \in \mathbb{C}} \text{\# occur. of } w' \text{ preceding } w > 0.$$

Similarly, for each word type w, if we consider its following words as contexts, we can compute its frequency as follows:

$$tokenC_{following}(w, \mathbb{C}) = \sum_{w' \in \mathbb{C}} \text{\# occur. of } w' \text{ following } w.$$

When we consider the preceding word as a word's context in sequence, word frequency can also be computed as

$$tokenC_{preceding}(w, \mathbb{C}) = \sum_{w' \in \mathbb{C}} \text{\# occur. of } w' \text{ preceding } w \ .$$

We now have four kinds of diversity measurements, by which we can rank the word types and distinguish the top words from others. Note that when there is no constraints imposed on which words are justified as proper contexts in computation, measurement $tokenC_{following}(w, \mathbb{C})$ equals $tokenC_{preceding}(w, \mathbb{C})$, both of which computes word frequency $tokenC(w, \mathbb{C})$. Thus, in Table 2.1, we present three lists of words: the first column contains the top 100 most frequent words in the WSJ corpus, i.e. ranked by $tokenC(w, \mathbb{C})$; the second column contains the top 100 words ranked by $typeC_{following}(w, \mathbb{C})$; and the third one contains the top 100 words ranked by $typeC_{preceding}(w, \mathbb{C})$. As shown in Table 2.1, the top 10 words of all three lists are heavily used closed-class words (and interestingly, the top 8 words of each list constitute a permutation of each other). Going down to the bottoms of these lists, more and more words fall in the open-class category, but the second column, ranked by $typeC_{following}(w, \mathbb{C})$, seems contain more closed-class words. If we assume

$tokenC$	$typeC_{following}$	$typeC_{preceding}$
the of to a	the and of a	and in to of
in and for that	to in for that	the for a that
is it **said** on	its by is with	is on by **million**
at by as from	from are as was	at as from with
with **million** was be	on be at or	was **said** has will
its are he but	their an his were	are or it would
has an have will	but have it has	**billion** have had its
new or **company** they	been **new said other**	were an about this
this which **year** would	some which **more** this	because **more** their also
about **market says more**	also about will he	into could over his
were had **billion** their	not had **one** who	out up after **last**
his up **one** than	they **says** than would	may who when than
stock been some who	any up **market** all	can **market shares business**
also **other share** not	**most two** no **million**	some **says** through **such**
we when **last** if	**many first only company**	under **only** if **new**
i all **shares president**	**major** after **such** when	**group** them before during
years trading first two	into can while could	**sales** all **one** they
after because could **sales**	these those now another	but **company** since **yesterday**
out there do **only**	**big** still her so	against now he **companies**
business such most can	**own** our **government** even	**stock** between **prices** any
york into may over	**business** both may being	**program** did should so
group many time now	like just if down	**two products people next**
federal companies prices no	**including** we i over	**stocks trading investment** down
government so any **cents**	**billion** off through **american**	until do off **rose**
quarter bank investors down	between **recent three make**	does back **operations officials**

Table 2.1: Top 100 words in the WSJ corpus ranked by different diversity measurements (open-class words are printed in bold).

theses lists as closed-class words, for all three columns, commonly used nouns constitute the majority of errors. Fairly speaking, it is not convincing to take any list in Table 2.1 as a closed-class lexicon, therefore, we need a better mechanism to acquire closed-class words other than by ranking only. In the next section, we are going to show that, we need to measure contextual diversities according to proper contexts only, and in Section 2.4.3, we are going to show how these ideas are manifested in a bootstrapping algorithm.

2.4.2 The proper contexts

It is proposed in (Chan, 2008) that morphological transformations should be discovered with respect to 'base forms', i.e. regarding properly justified word stems only. In this work, we generalize this idea to measure contextual diversities of closed-class words as well. More specifically, we would like to acquire a set of 'justified' contexts, and compute contextual diversities of the words to be classified according to the justified contexts only. We consider a word as a proper context for the words to be classified, only if it has ever been seen as context of more than one (already recognized) closed-class words. For example, suppose that we consider the following word as a word's context. Assume that we have already acquired words *the* and *a* as closed-class words, then only the set of words that have ever been seen following both of them should be considered as justified contexts while measuring contextual diversities of other word types to be classified. More formally, given a set of justified contexts $\mathbb{B}$, as well as a corpus $\mathbb{C}$, we re-write the four types of diversity

measurements as follows:

$$typeC_{following/preceding}(w, \mathbb{B}) = \sum_{w' \in \mathbb{B}} \text{\# occur. of } w' \text{ following/preceding } w > 0;$$

$$tokenC_{following/preceding}(w, \mathbb{B}) = \sum_{w' \in \mathbb{B}} \text{\# occur. of } w' \text{ following/preceding } w \ .$$

When the set of justified context words does not contain all the word types in corpus, measurement $tokenC_{following}(w, \mathbb{B})$ no longer equals $tokenC_{preceding}(w, \mathbb{B})$.

Given a diversity measurement function div for ranking closed-class words, we can use the symmetric type-based measurement div' to compute set $\mathbb{B}$ that contains justified contexts. When the measurement function $div = tokenC_{following}$ or $typeC_{following}$, the justified contexts contain words that have ever been seen following more than one words in the acquired output, i.e. the symmetric measurement function for context words is $div' = typeC_{preceding}$. When the measurement function $div = tokenC_{preceding}$ or $typeC_{preceding}$, the justified contexts contain words that have ever been seen preceding more than one words in the acquired output, i.e. the symmetric measurement function for context words is $div' = typeC_{following}$. More formally, given a set of acquired words $\mathbb{F}$, and a type-based measurement function div', the set of justified contexts $\mathbb{B}$ consists of those elements that has a type-based diversity greater than one, i.e.

$$\mathbb{B} = \{w | div'(w) > 1\}.$$

To make a coherent reference to the two kinds of contextual directionality, we introduce some notations here. If an *in*-context relationship holds between two adjacent words (w_i, w_{i+1}), then there is a *as*-context relationship holds between (w_{i+1}, w_i). Symmetrically,

if an *in*-context relationship holds between two adjacent words (w_{i+1}, w_i), then there is a *as*-context relationship holds between (w_i, w_{i+1}). In other words, if the following word is considered *as*-context of the preceding word, then the preceding word is considered *in*-context of the following word. Symmetrically, if the preceding word is considered *as*-context of the following word, then the following word is considered *in*-context of the preceding word. Given these notations, we rank the words by measurement $token/typeC_{\textit{in}\text{-context}}$ to distinguish closed-class words, and correspondingly compute the set of justified contexts by measurement $typeC_{\textit{as}\text{-context}}$. Therefore, we can view the set of acquired output $\mathbb{F}$ and the set of justified contexts $\mathbb{B}$ as two complementary sets, both of which justify proper contexts for each other. So as to iteratively generate these two complementary sets, we propose a bootstrapping algorithm as described in the next section.

2.4.3 The bootstrapping algorithm

The proposed bootstrapping algorithm in Algorithm 1 generates two complementary sets during the bootstrapping process, both of which justify proper contexts for each other to compute contextual diversity . As the two complementary sets updated during the bootstrapping process, the diversity measurements of the items in the other set are expected to be more and more accurate. Inputs to this algorithm are a dataset of words $\mathbb{S}$ and a pair of diversity measurements as defined in the above section. For each word in the input dataset, the number of its occurrences following/preceding other words are also provided.

Based on the idea that closed-class words can be distinguished by higher contextual

Algorithm 1 The bootstrapping algorithm for acquiring functional elements

Require: a data set $\mathbb{S}$ to be classified

Require: a pair of diversity measurement functions div and div'

initialize set $\mathbb{F}$ and set $\mathbb{R}$ to be empty and initialize set $\mathbb{B}$ to be $\mathbb{S}$

for k iterations **do**

let $\mathbb{F}$ be the top k most diverse elements with respect to $div(e, \mathbb{B})$, for e in $\mathbb{S} - \mathbb{R}$

let $\mathbb{B}$ be those elements with a diversity greater than one, i.e. $div'(e, \mathbb{F}) > 1$, for e in $\mathbb{S}$

let $\mathbb{R}$ be those elements too often *as/in*-context of any element in $\mathbb{F}$

end for

return $\mathbb{F}$ and $\mathbb{B}$

diversity, we explicitly let a set $\mathbb{F}$ contain the most diverse elements ranked by the chosen measurement with respect to its complementary set $\mathbb{B}$. More specifically, at the $k + 1$ iteration, we recompute the acquired set F_{k+1} of the top $k + 1$ words according to the set of justified contexts B_k. And the set of justified contexts B_k is computed according to the acquired output F_k from the k iteration, which contains top k most diverse words. Since the ranking order of words varies over iterations, with respect to the update of justified contexts, a word that is acquired into $\mathbb{F}$ at some iteration is not guaranteed to be acquired into $\mathbb{F}$ in the following iterations. In addition, since we are generating two complementary sets by this bootstrapping algorithm, we do not want these two sets overlap too much. In other words, if a word is often seen *as*-context of already acquired closed-class words, it should not be considered as acquired output in the following iteration. Symmetrically, we also filter out those words that are too often seen *in*-context of already acquired closed-class

words. More specifically, we maintain a set $\mathbb{R}$ containing elements to be excluded from the acquired closed-class set. After the update of $\mathbb{F}$ at each iteration, we update the filtering set $\mathbb{R}$ with those words r that are too often seen *in/as*-context of acquired words in $\mathbb{F}$, i.e. given a threshold number THR, $tokenC_{\textit{in/as}\text{-context}}(r, \mathbb{F})/tokenC(r) > THR$.

Since it is very common to see two closed-class words in adjacency, it seems no harm to let two words in the acquired output form *in*-context relationships. By imposing the constraint that no two acquired words form *in*-context relationships we are actually acquiring 'first-order' closed-class words such as determiners and auxiliary verbs. In a morphologically poor language such as English, it is the first-order closed-class words that bare the syntactic roles that may also be reflected with morphological transformations. And in a language observing no morphology, such as Chinese, it is the first-order closed-class words that are also put silent. Even though the linguistic idea behind the filtering step is too vague to be clearly presented, this step is finally kept in the proposed algorithm due to the improvement of performance it introduces to our experiments.

We also need to decide when the bootstrapping stops. Theoretically, we would like to stop the bootstrapping process when the last word w_k of the acquired list doesn't show in diverse enough contexts, i.e. given a measurement div and a threshold d, the bootstrapping stops when $div(w_k) < d$. In practice, for the sake of comparison, we simply let the bootstrapping stops after 25 iterations. As shown in Table 2.2, by all four types of diversity measurements, most of the acquired output fall in the closed-class category. However, only with the type-based measurement regarding the following word as context, i.e. $typeC_{following}$, the acquisition output are actually 'first-order' closed-class words, i.e.

determiners, possessive pronouns and modal verbs in English.

2.5 The acquisition of morphological transformations

In this section, we are going to apply Algorithm 1 to acquire morphological transformations, especially morphological stems and suffixes for English, thus the proposed bootstrapping algorithm is applied to acquire both free and bound forms of functional elements. We are aware of the bulk of literature on morphological learning and in this section, we merely focus on the application of a general bootstrapping algorithm to this classic problem. In Chapter 3, we will dig this problem further and make more comments on the literature.

The first key idea behind the proposed algorithm is that functional elements, including morphological transformations and closed-class words, tend to occur in more diverse contexts than other elements. As in the case of morphology learning, we expect to see that morphological endings can be combined with various types of stems to form legal words, but arbitrary suffixes can be found in few word types only. For example, an inflectional ending *'-ed'* can be concatenated to most verb stems to derive past tense forms, compared to which a non-sense suffix *'-roached'* can only be seen in few particular word types.

Second, we still have options to choose between token-based or type-based diversity measurements, but contextual directionality doesn't bother any more. For any division of a legal word, the stem is considered *as*-context of the suffix and the suffix is considered *in*-context of the stem. Similar with the acquisition of closed-class words, the baseline model is not very sensitive to the choice of token-based or type-based measurements. For

diversity	acquistion output				
$tokenC_{following}$	the[48759] ,	a[18889] ,	**it**[5641] ,	**he**[3842] ,	its[3519]
	an[3080] ,	this[2603] ,	**they**[2540] ,	their[1737] ,	**who**[1705]
	his[1691] ,	some[1614] ,	**we**[1246] ,	**i**[1134] ,	**there**[955]
	no[869] ,	any[858] ,	**york**[812] ,	**you**[732] ,	these[601]
	she[559] ,	**according**[497] ,	several[437] ,	our[406] ,	another[402]
$tokenC_{preceding}$	the[43826] ,	a[18596] ,	**it**[4715] ,	its[4071] ,	an[2967]
	this[2002] ,	**they**[1850] ,	their[1849] ,	his[1755] ,	**he**[1657]
	some[1266] ,	**york**[1118] ,	any[857] ,	**no**[710] ,	**we**[676]
	there[622] ,	**you**[571] ,	those[560] ,	**i**[470] ,	these[433]
	officer[394] ,	another[377] ,	several[375] ,	our[350] ,	**due**[326]
$typeC_{following}$	the[3143] ,	a[1989] ,	its[1188] ,	their[792] ,	his[706]
	some[575] ,	will[470] ,	an[444] ,	any[430] ,	would[401]
	can[299] ,	could[298] ,	these[282] ,	those[278] ,	our[275]
	another[269] ,	may[225] ,	her[223] ,	several[185] ,	**york**[181]
	my[155] ,	might[136] ,	each[133] ,	whose[133] ,	your[129]
$typeC_{preceding}$	**in**[2679] ,	**to**[2452] ,	**of**[1955] ,	**for**[1806] ,	**on**[1382]
	by[1213] ,	**from**[1088] ,	**with**[1058] ,	would[678] ,	could[451]
	after[405] ,	**into**[399] ,	can[395] ,	**under**[315] ,	**before**[299]
	during[286] ,	**since**[272] ,	**against**[234] ,	**says**[189] ,	might[185]
	without[180] ,	**among**[149] ,	**wo**[125]		

Table 2.2: Acquisition output of the bootstrapping algorithm over the WSJ corpus. Those acquired words that are not considered as first-order closed-class words are printed bold. Contextual diversities of each acquired item at the final iteration are given in [].

example, the top three most frequent suffixes in the WSJ corpus are *'-s'*, *'-d'* and *'-e'*, and the top three suffixes that occur in most word types happen to be *'-s'*, *'-d'* and *'-e'* as well.

The last key idea behind the bootstrapping algorithm is even more crucial for morphology learning: contextual diversity should be measured according to justified stems only. During the acquisition process, the most simple way of justifying a stems as a proper context is to check whether it forms legal words with more than one of the acquired suffixes. For example, when suffixes *-ed* and *-s* have been acquired as morphological suffixes, stem *lie-* should not be justified as a proper context yet, given that it can only form legal words with a single suffix *-s*; but stem *laugh-* can be considered as a proper context, since it can form legal words with more than one acquired suffixes. When suffix *-d* is also acquired as a morphological suffix during the bootstrapping process, stem *lie-* can then be justified. This idea of measuring contextual diversity regarding proper contexts only relates to the use of 'base form' for morphology learning in (Chan, 2008), but in a more abstract way.

More formally, given a set of justified stems B and a corpus $\mathbb{C}$, we measure the type-based contextual diversity of a suffix f as

$$typeC(f, \mathbb{B}) = \sum_{t \in \mathbb{B}} \mathbb{1}_{\mathbb{C}}(t.f),$$

where $\mathbb{1}_{\mathbb{C}}(t.f)$ is set to 1 if $t.f$ forms any legal word in $\mathbb{C}$, otherwise 0. For example, if we are given a set of justified stems, including *'laugh-'* but not *'b-'*, the diversity measurement of *'-ing'* will increase by one given the existence of word *'laughing'* but not by the existence

of word *'bing'*. Similarly, the token-based contextual diversity can be computed as follows:

$$tokenC(f, \mathbb{B}) = \sum_{w \in \mathbb{C}} \mathbb{1}_{\mathbb{B}}(w = (t.f)),$$

where $\mathbb{1}_{\mathbb{B}}(w = (t.f))$ is set to 1 if $w = (t.f)$ for any stem t in $\mathbb{B}$, otherwise 0. We also need to update the set of justified stems B at each bootstrapping iteration. Given a set of acquired suffixes F, we measure the contextual diversity for a stem t as $typeC(t, \mathbb{F}) = \sum_{f \in \mathbb{F}} \mathbb{1}(t.f)$. As the same with the acquisition of closed-class words, when $typeC(t, \mathbb{F}) > 1$, t can be justified as a proper context.

Recall that, for the acquisition of closed-class words, set $\mathbb{B}$ is initialized as the whole word set at the beginning of bootstrapping, which is also the fixed set of contexts for the baseline models. For morphology learning, we try two different sets for initialization: set $\mathbb{B}^1$ that contains all possible suffixes and set $\mathbb{B}^2$ that contains legal words as proper stems only. As shown in Table 2.3, bootstrapping with measurement $typeC$ is also the best model for morphological learning, and this model performs stably with different initializations.

2.6 Experiments on acquiring a closed-class lexicon

Our proposed bootstrapping algorithm is applied to acquire the first-order closed-class words in Section 2.4, and applied to acquire morphological endings in Section 2.5. Suppose that we acquire the first-order closed-class words with diversity measurement function div. Both morphological and word-based acquisition outputs can be put together to build a closed-class lexicon in the following way:

alg.	div.	init.	output at the 10th iteration
baseline	tokenC	$\mathbb{B}^1$	'*-e*', '*-s*', '*-t*', '*-d*', '*-n*', '*-he*', '*-r*', '*-y*', '*-o*', '*-ed*'
		$\mathbb{B}^2$	'*-s*', '*-d*', '*-n*', '*-nd*', '*-t*', '*-ed*', '*-e*', '*-ing*', '*-y*', '*-.*'
	typeC	$\mathbb{B}^1$	'*-s*', '*-e*', '*-d*', '*-ed*', '*-g*', '*-n*', '*-ng*', '*-y*', '*-ing*', '*-t*'
		$\mathbb{B}^2$	'*-s*', '*-ing*', '*-ed*', '*-d*', '*-ly*', '*-er*', '*-5*', '*-0*', '*-y*', '*-e*'
bootstrap	tokenC	$\mathbb{B}^1$	'*-e*', '*-ed*', '*-s*', '*-f*', '*-ing*', '*-es*', '*-r*', '*-ion*', '*-or*', '*-at*'
		$\mathbb{B}^2$	'*-s*', '*-n*', '*-t*', '*-f*', '*-nd*', '*-r*', '*-re*', '*-ll*', '*-''*', '*-d*'
	typeC	$\mathbb{B}^1$	'*-ed*', '*-ing*', '*-e*', '*-es*', '*-s*', '*-er*', '*-ers*', '*-ion*', '*-ions*', '*-y*'
		$\mathbb{B}^2$	'*-ed*', '*-ing*', '*-e*', '*-es*', '*-s*', '*-er*', '*-ers*', '*-ion*', '*-ions*', '*-y*'

Table 2.3: The acquisition outputs over WSJ corpus. Set $\mathbb{B}^1$ contains all possible suffixes and $\mathbb{B}^2$ contains legal words as proper stems only.

rank all the word types by div, and from the word with highest diversity, go through this list until we get enough words for the lexicon:

- if a word is acquired as first-order closed-class, thrown to the lexicon ;
- if a word contains an acquired morphological ending (with the corresponding stem of a reasonable length), then excluded from the lexicon;
- if a word occurs too often *as*-context of acquired first-order closed-class words, then excluded from the lexicon;

- otherwise, add the current word to the lexicon.

In the following experiments, we varies the acquisition model for closed-class words, but stick to the morphological model that produces the best result as shown in Table 2.3, i.e. the one with type-based diversity measurement.

So as to evaluate the acquired lexicons, we collected a gold closed-class lexicon from the WSJ corpus with the help of POS annotations. The gold lexicon contains 288 words whose primary POS category is closed-class. More specifically, we consider 6 open-class categories and each of them covers a set of Penn Treebank POS tags considered as open-class, and all other tags are considered as falling in the closed-class category.

- Nouns: ('NN', 'JJ','JJS','JJR','NNS','NNP','NNPS','FW','LS','SYM', 'UH');
- Verbs: ('VB', 'VBD','VBP','VBZ');
- Present participle: ('VBG',);
- Past participle: ('VBN',);
- Adverbs:('RB' 'WRB','RBR','RBS');
- Numbers: ('CD',).

Given the gold lexicon containing 288 words, we can now compute $fscore$ for each acquired lexicon. Let $goodc$ be the count of the overlap words of the acquisition output and the gold lexicon, let $errc$ be the count of words in the acquisition output that are not in the gold lexicon, and let $missc$ be the count of words that are in the gold lexicon but not in the acquisition output. Then we compute $fscore$ as follows:

$$prec = \frac{goldc}{errc + goldc} \qquad recall = \frac{goldc}{missc + goldc} \qquad fscore = \frac{2 * prec * recall}{prec + recall}$$

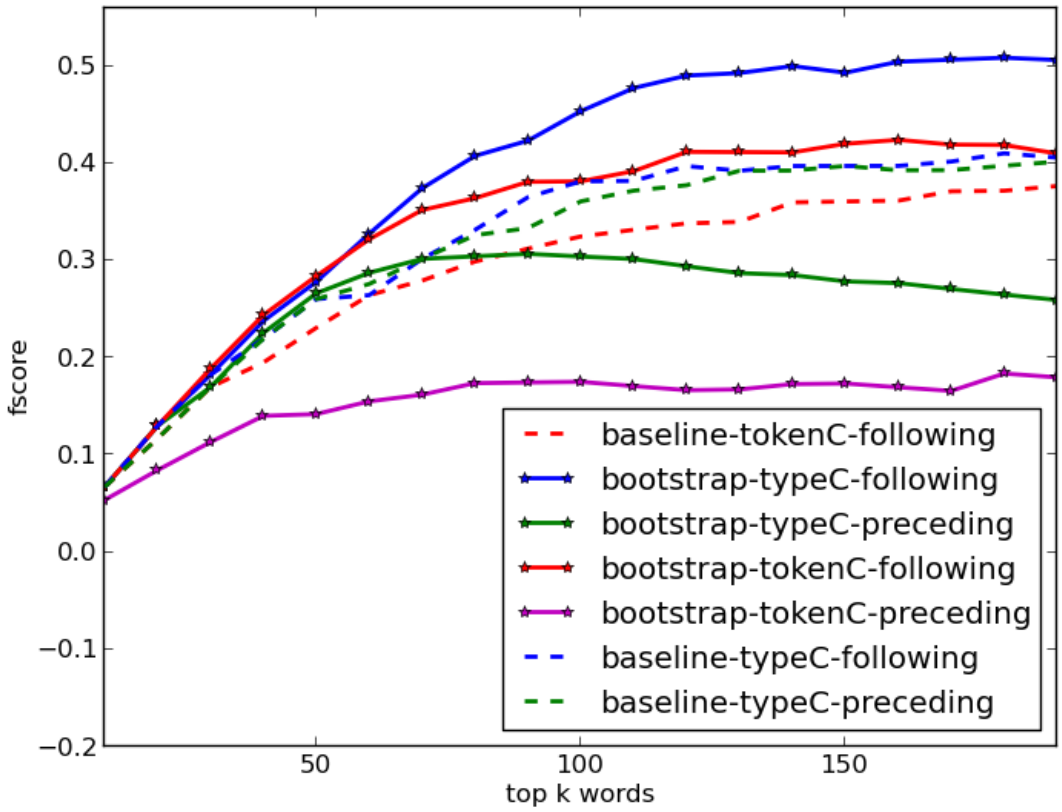


Figure 2.1: The $fscore$ of each acquired lexicon with respect to different lexicon sizes.

In Figure 2.1, we plot $fscore$ with respect to lexicon sizes. As discussed above, we fix the morphological model to build the closed-class lexicon but vary the acquisition model for first-order closed-class words. The lines in Figure 2.1 thus correspond to the first-order acquisition output in Table 2.2. For the baseline models, type-based diversity measurements introduces slightly fewer errors in output. However, the bootstrap algorithm is much more sensitive to the choice of diversity measurement. With this experiment, we can draw a

algorithm	measurement	token-based acc.
baseline	tokenC	87.01%
	typeC	88.83%
bootstrap	tokenC	**92.37%**
	typeC	90.74%
(Graça 2011)		92%

Table 2.4: Token-based evaluation of the close- vs. open- class distinction.

preliminary conclusion that our proposed bootstrapping algorithm generates better outputs than the baseline model when the choice of diversity measurement is proper.

We report a type-based evaluation in Figure 2.1, and in Table 2.4, we evaluate the acquisition outputs with token-based accuracy. With a token-based evaluation, word frequencies are weighted in and predictions on frequent words affects results more. More specifically, to perform a token-based evaluation, we consider a tagging problem with a tagset of two tags: closed- vs. open-class. The gold POS annotations of the WSJ corpus are transformed to binary tags according to our classification on the POS tags described above. Tagging predictions are made by looking up the acquired closed-class lexicon: if a word is in the closed-class lexicon, the closed-class tag is assigned to each of its occurrences; otherwise, an open-class tag is assigned. The tagging accuracy is then computed as percentage of correct predictions out of all words, as usual. In Table 2.4, we stick to the context of a word as its following word, since as shown in Figure 2.1, models that assume the context

of a word as its preceding word perform the worst. Then both the bootstrapping algorithm and the baseline counting varies their diversity measurement functions, either type-based or token-based. It is worth special attention that, even though with the type-based evaluation (Figure 2.1), the bootstrapping model with a type-based diversity measurement performs the best, with a token-based evaluation, it is the bootstrapping model with a token-based diversity measurement that performs the best. Therefore, it is important to try different evaluation criteria, since different aspects of alternative models may be better revealed in different experiments. The results of our models are compared to the token-based accuracy reported in (Graça, 2011), which acquires close- vs. open- class distinction in one language from annotated corpora of other language. Our bootstrapping model requires much less resource, but achieves the same level of performance. In section 2.8, we are going to plug the acquired lexicons into a two-stage POS tagging system (Section 2.7) and obtain a totally unsupervised tagging model achieving very promising performance.

2.7 Functional elements and POS tagging

In this section, we propose a new model for unsupervised POS tagging based on the linguistic distinction between open- and closed-class items. In a supervised POS tagging task, a tagger is trained on labeled sequences; however, in an unsupervised POS tagging task, there is no gold predictions to learn from.

As in the context of this work (Zhao and Marcus, 2009), it is a common assumption for an unsupervised POS tagging system to take a dictionary or a partial dictionary as given,

which specifies possible tags for all or some of the word types. In previous work such as (Smith and Eisner, 2005) and (Goldwater and Griffiths, 2007), the tagging performance always goes down very quickly as the size of the given dictionary goes down. Therefore, one of our contributions by this work is reducing the requirement of large input dictionaries by unsupervised taggers, which require either expertise or annotated copora to build. Provided with only a closed-class lexicon of 288 words, about 0.6% of a full lexicon, our tagging system first acquires a large open-class lexicon and then acquires disambiguation rules for both closed- and open-class words, achieving a tagging accuracy of 90.6% for a 24k dataset, not far from the state-of-art performance (93.4%) achieved with a full dictionary (Toutanova and Johnson, 2008).

2.7.1 A feature-based analysis of POS tags

All recent research on unsupervised tagging, as well as the majority of work on supervised taggers, views POS tagging as a sequence labeling problem and treats all POS tags as equivalently meaningless labels. However, the engineering concept of POS tags actually derives from the linguistic notion of syntactic category which specifies the combinatorial properties of a word in an underlying (syntactic) structure. For example, When an occurrence of word *composed* is tagged by 'VBD' (past tense), we know that it can take arguments as a verb in this context; however, if another occurrence of the same word is tagged by 'VBN' (past participle), we know that it functions as adjectives in this context.

With either tag 'VBN' or 'VBD', a word is labeled as semantically verbal, thus the

lexical information of the word itself doesn't suffice to disambiguate these two tags. The feature that distinguishes between 'VBN' and 'VBD' is whether the word is syntactically nominal or not as imposed by the local context. For example, if we see an occurrence of word *composed* following a determiner, we know that it is functioning as nominals, so it should be tagged by 'VBN'; however, if we see another occurrence of the same word following a subject pronoun, we know that it is functioning as a verb, so it should be tagged by 'VBD'. Motivated by this observation, we propose a feature-based analysis of 6 open-class categories as depicted in Table 2.5, each of which covers a set of POS tags.

	Nominal	Verbal	inflection
Nouns	+	-	('-', '-s', '-er', '-est')
Verbs	-	+	('-', '-s', '-ed', '-ing')
Past participles	+	+	('-ed',)
Present participles	+	+	('-ing',)
Adverbs	-	-	('-',)
Numbers	+	-	('[0-9.]+',)

Table 2.5: A feature-based analysis of the open-class categories.

As shown in Table 2.5, we have introduced two binary features:

- Syntactically nominal or not in local context, to be disambiguated in tagging.
- Semantically verbal or not, to be induced by distributional clustering.

Since POS tags in Penn Treebank tagset are encoded with inflectional information as well,

an inflection feature is also introduced in our analysis. Given such a feature-based analysis, the tagging task can be decomposed into subtasks setting these features.

Whether a word is semantically verbal or not doesn't depend on the local context it occurs in. Setting this feature is equal to inducing two lexical categories: verbal and nominal. Following previous work on POS induction, e.g. (Clark, 2003) and (Schütze, 1993), we also base our work on distributional clustering. Combining this binary clustering with inflection feature together, we can compute possible POS tags for each open-class word. We will discuss more about acquiring the open-class lexicon in Section 2.7.3. In contrast, whether an occurrence of a word is syntactically nominal or not is imposed by its local context. Instead of formulating it as a sequential labeling problem with a tagset of two tags, we propose to set this local feature by a rule-based disambiguation model. In Section 2.7.4, we show the advantage of using such a simple model for unsupervised tagging.

2.7.2 The two-stage system design

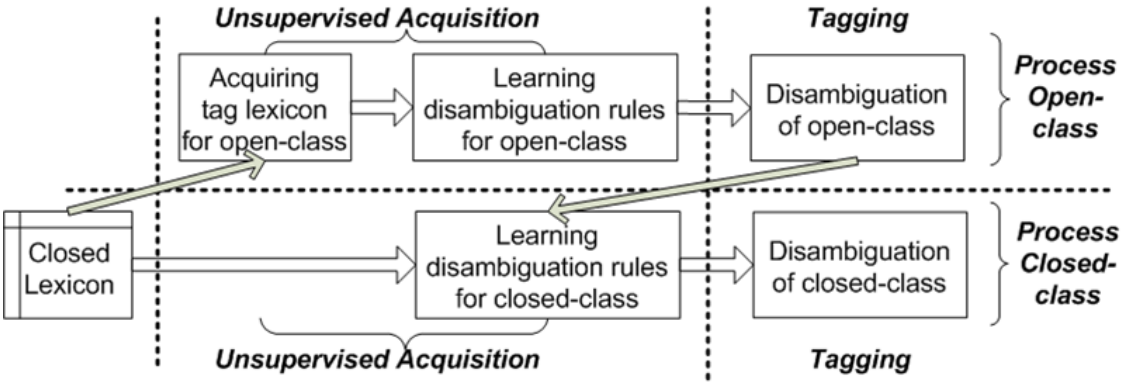


Figure 2.2: The two-stage unsupervised tagging system.

As one may have noticed, along with introducing a feature-based analysis of open-class POS tags, we have drafted a plan for acquiring open-class lexicon and learning the disambiguation model. On the other hand, closed-class tags are much more ad-hoc than open-class tags, and many of them label a couple of words only, such as 'EX' for *there*, 'DT' for determiners and so on. In this work, we assume the input of a closed-class lexicon, the gold one described in Section 2.6. This closed-class lexicon tells apart closed-class words from open-class words, and specifies all possible POS tags (without frequency information) for each closed-class word. Furthermore, we acquire the disambiguation model for closed-class words with a similar learning schema as for open-class words.

Overall, based on the distinction between open- and closed-class words, we propose a two-stage unsupervised tagging system. First, we acquire possible tags for each open-class words by distributional clustering and learn a rule-based model to disambiguate open-class words. Both of these tasks require the input of a closed-class lexicon to specify the distinction between open- and closed-class. Second, with the help of the acquired open-class tagger, we learn another rule-based model to disambiguate closed-class words. In Figure 2.2, we depict the structure of this two-stage unsupervised tagging system.

2.7.3 Acquiring open-class lexicon

Inducing lexical categories is a language acquisition task on which there has been extensive research. Following a classic work (Schütze, 1993), we also base our work on distributional clustering. More specifically, each word type, w, is represented by a feature vector of

length m, i.e. $<$*count*(w,C_1),...,*count*(w,C_i),...,*count*(w,C_m)$>$, where each component of the feature vector is a count of the occurrences of w in context C_i. These feature vectors are then to be clustered according to similarity.

As is well known, distributional clustering of all word types results in a high number of clusters, for example, Schütze (1993) induces 200 clusters. Most of these induced categories are difficult to associate with a specific POS tag, therefore, Chan (2008) argues that the restriction to cluster base forms onlyis crucial to induce clusters more in line with lexical categories, i.e. the open-class categories we care about here. As discussed in Section 2.5, base forms are justified contexts for morphological transformations; however, in this application, we simply extract base forms by stripping three inflectional endings, *-s*, *-ing* and *-ed* from open-class words. Furthermore, in previous work, the contextual features coded in clustering vectors are usually lexical, so a typical feature vector contains hundreds to thousands of components. A chosen clustering algorithm then runs over this high-dimensional space, thus computationally quite intensive. Instead, we propose to represent base forms by functional elements in their local contexts only, and each base form is represented by a feature vector of five components:

- the count of preceding determiners in all its occurrences,
- the count of following determiners in all its occurrences,
- the count of its *-ed* inflections in the whole corpus,
- the count of its *-ing* inflections in the whole corpus, and

- the count of its *-s* inflections in the whole corpus.

This radical reduction of the feature space enables a substantial improvement in efficiency, but doesn't hurt the performance at all. Also for such a simple clustering task, there is no need resorting to sophisticated techniques. We use a basic k-means clustering algorithm which allows us to specify the number of output clusters (Maffi, 2007).

Combining this binary clustering with inflection features, we can compute possible tags for each open-class word now, as depicted in Figure 2.3. For example, if the base form *start* is classified into the verbal class, then both its inflections *starts* and *start* will receive one possible tag 'VB'; its inflection *starting* will receive one possible tag 'VBG'; but its inflection *started* will receive two possible tags 'VBN' and 'VB'. In this example, the nominal senses of *start* and *starts* are missing. For such cases, we introduce a simple supplemental process : if a base form is ever seen following a determiner, it will be included in the nominal class as well.

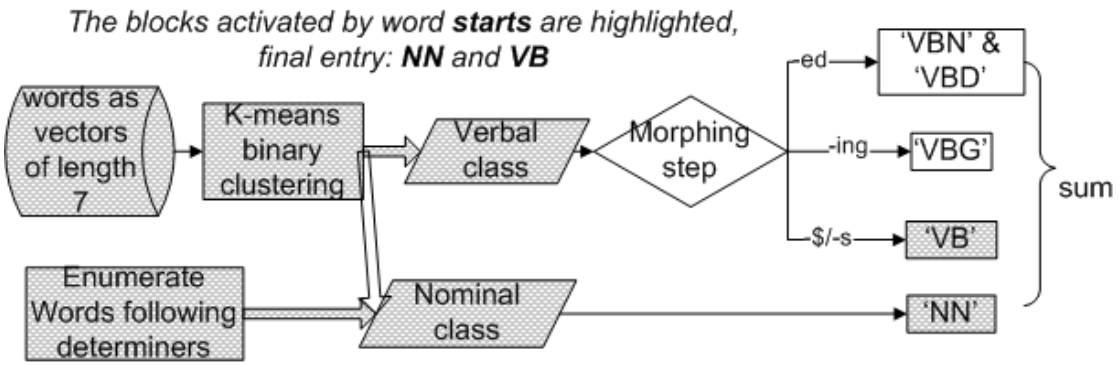


Figure 2.3: Compute possible POS tags for each open-class word.

2.7.4 Unsupervised POS Tagging

Taking a dictionary as input, the task of unsupervised POS tagging is to learn a disambiguation model from unannotated data and then use this model for decoding. We will first introduce our design of the disambiguation model for open-class words and then the corresponding model for disambiguating closed-class words.

Disambiguation model for open-class words

Given the feature-based analysis of open-class POS tags, we only need to make a binary decision for the disambiguation of open-class words: whether the word is imposed syntactically nominal or verbal by the local context. Therefore, we propose a rule-based disambiguation model with each rule conditioned on functional contexts, and predicts the Nominal/Verbal category imposed by this context. More specifically, each disambiguation rule is written as $r = (con : cat)$, with con and cat the functional context and categorical information (*N/V*) respectively. While disambiguating an open-class word, functional context con is checked against the preceding closed-class word (if any), and N/V category cat of the following open-class word is predicted. For example, a disambiguation rule for open-class words, *he:V*, says that if an open-class token follows the closed-class item *he*, then a verbal tag should be assigned to this token.

Although there is no annotated data available for learning, we can use the unambiguous occurrences in data to establish disambiguation rules and apply the rules to ambiguous occurrences. For open-class words, disambiguation rules are extracted from raw data. A

pair of adjacent words (W_l, W_r) is considered observing an unambiguous application of a disambiguation rule if it satisfies the following two conditions: 1. the context word W_l falls in the closed-class category; and 2. all possible tags of the current word W_r fall in the same N/V category (Nominal or Verbal but not mixed). If (W_l, W_r) is unambiguous in this sense, then extract rule $r = (con : cat)$, where con is W_l, and cat is the N/V category of W_r. For example, in the sequence *(...he has claimed..)*, the pair of adjacent words *(he, has)* can be used in that *he* is a closed-class item and *has* has only one possible tag, 'VB', so a rule $(he : V)$ is extracted; but *(has, claimed)* is not usable since *claimed* has two incoherent possible tags: 'VB' of category V and 'VBN' of category N.

In a counting step, a set of rules R is first initialized to be empty, and then, as each disambiguation rule r is generated while passing through the data, if not already in R, it is added with an initial count of one; otherwise, N_r, the count of rule r, is increased by one. Since we know that for a rule, $(con : cat)$, the prediction cat can only be either N or V, thus for each context con, there are only two forms of rules counted, $(con : N)$ or $(con : V)$. After the counting step, we select the rule with a greater count for each context, thus guarantee that the resulting disambiguation model is deterministic.

Given our rule-based, deterministic disambiguation model, tagging is a straightforward process decoding the rules. For each ambiguous open-class word w in sequence if the preceding closed-class word (if any) invokes a disambiguation rule, $r = (con : cat)$, then pick a possible tag of w that falls in cat (N or V). Since each open-class word may have mostly one tag in either *N/V* category, as discussed in Section 2.7.3, any invoked disambiguation rule gives a deterministic result. If no rule is triggered then our default choice is 'NN'; but

if 'NN' is not a possible tag for the current word, we assume the local domain is verbal. In Figure 2.4, we give a couple of examples of tagging open-class words.

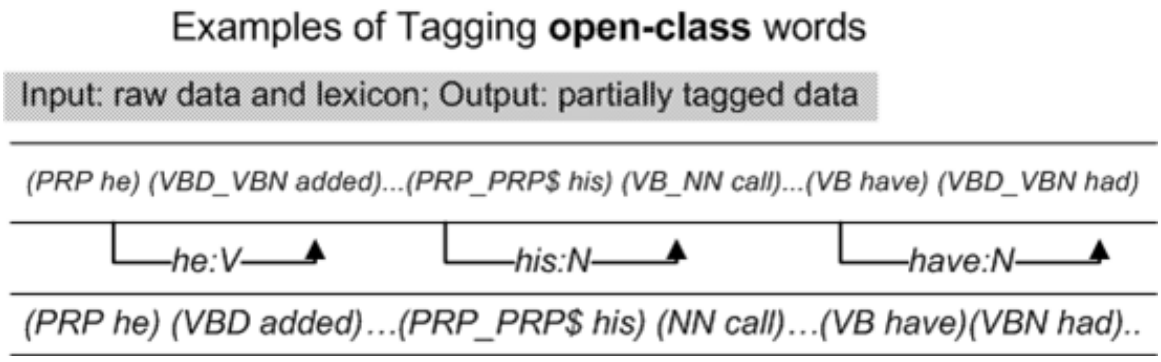


Figure 2.4: Examples of tagging open-class words.

Disambiguation model for closed-class words

As depicted in Figure 2.4, after tagging open-class words, besides raw data, we also have open-class tags available to help learning disambiguation models for closed-class words.

The disambiguation rules for closed-class words are conditioned on Nominal/Verbal categories and predicts closed-class POS tags, thus in exactly the symmetric form of the disambiguation rules for open-class words. After all open-class words are tagged, a pair of adjacent words (W_l, W_r) is considered observing an unambiguous application of a closed-class disambiguation rule, if it satisfies the following two conditions: 1. the current word W_l is in the closed-class lexicon and has only one possible tag; and 2. the context word W_r is either open-class (thus already tagged) or having all possible tags fall in the Nominal

category[2]. If a pair (W_l, W_r) is unambiguous in the above sense, then extract rule $r = (con : cat)$, where the prediction cat is the single POS tag of the current word W_l, and context cat is the N/V category of the context word W_r. For example, in the sequence *"...for his stepping..."*, the pair *(for his)* is unambiguous in that *for* has only one possible tag 'IN' and both possible tags of *his*, 'PRP' and 'PRP$', fall into the Nominal category, so a rule $(N : IN)$ is extracted; but *(his stepping)* is not usable since *his* has more than one possible tag even though *stepping* is already tagged.

By selecting the rule with a greater count for each context, we guarantee that the resulting disambiguation model is deterministic, the same strategy as for open-class words. Thus the application of these deterministic rules are also very straightforward for disambiguating closed-class words. For each ambiguous closed-class word w in sequence, if it is followed by a word of category con (*N/V*), pick a possible tag of w, cat, such that $(con : cat)$ is a rule learned for disambiguating closed-class words. If no tag is picked, a random choice is made. We show some examples of tagging closed-class words in Figure 2.5. Even though there are residual cases where no functional context can help with tagging, the disambiguation strategy proposed here combined with random choices results in a good overall performance, as we are going to show in section 2.7.5.

[2]For Penn Treebank tagset, we consider 8 closed-class POS tags falling in the Nominal category: 'DT', 'PRP', 'PRP$', 'WDT', 'WP', 'WP$', '$', and '#', and all other closed-class POS tags has no category.

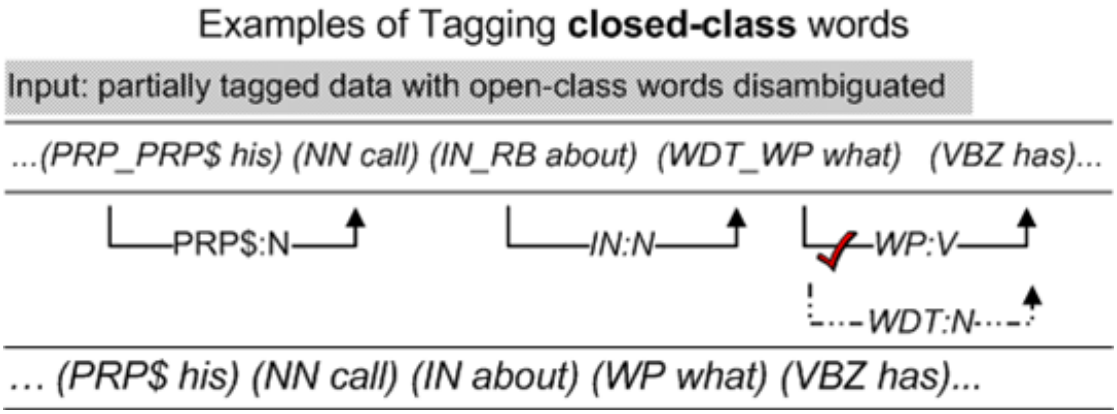


Figure 2.5: Examples of tagging closed-class words.

2.7.5 Experiemnts

As reported in (Banko and Moore, 2004), the quality of the lexicon made available to unsupervised learners made the greatest difference to tagging accuracy. Recall that, as described in section 2.6, our closed-class lexicon is automatically constructed from the WSJ corpus; therefore, we only compare our experiments to recent work built over automatically extracted lexicons from the same corpus. The proposed unsupervised tagging system is compared to the following models: CRF/CE (Smith and Eisner, 2005) which proposes contrastive estimation (CE) for log-linear models; BHMM2 (Goldwater and Griffiths, 2007) which uses Gibbs sampling for the inference of HMM-based generative models; and LDA+AC (Toutanova and Johnson, 2008) which extends the Latent Dirichlet Allocation model and incorporates the intuition that words' distributions over tags are sparse, achieving the current state-of-the-art performance.

As discussed in Section 2.6, we reduce the 22 open-class POS tags in Penn Treebank

tagset to 6 open-class categories, and in practice, we also ignore the difference between closed-class tags 'RP' and 'IN', or 'DT' and 'PDT'. As a result, our unsupervised tagging system runs over a reduced tagset of 27 tags. This is not unusual to use a reduced tagset for unsupervised tagging, and the models we are going to compare with use a reduced tagset of 17 tags (Smith and Eisner, 2005). In addition to reporting on our own tagset of 27 tags, we also map the tagging results onto the 17 tags used in other models for comparison[3].

Unsupervised POS tagging with partial dictionaries

	dict. with words of count $> d$				
d	1	2	3	∞	#tag
(part lex.)	(100%)	(55%)	(41%)	(0.1%)	-
BHMM2	87.3	79.6	65.0	-	17
CRF/CE	90.4	77.0	71.7	-	17
LDA+AC	**93.4**	91.2	89.7	-	17
our model	91.8	...	...	**90.6**	17
our model	*93.2*	...	...	*92.1*	27

Table 2.6: Tagging accuracy with partial dictionaries over 24k dataset; our closed-class lexicon is the closest approximation to the ∞ column .

[3]So as to map coarser categories to finer POS tags, random choices are made among all the possible POS tags corresponding to the coarser category. For example, tags 'RP' and 'IN' are reduced to one category in our tagset but kept distinct in the tagset of 17 tags; then for mapping the coarser category 'RP|IN' of a word, we need to make a random choice between 'RP' and 'IN'.

We use the same split of the dataset as previous work: the tagging model is trained over a 96k dataset and evaluated on a 24k dataset (Smith and Eisner, 2005). Again, tagging accuracy is calculated as the percentage of correctly tagged words out of all words in corpus. As shown in Table 2.6, if we want to accomplish the unsupervised tagging task with as little information as possible, reducing the dictionary by filtering rare words (with count$<= d$) has not been a promising track to follow. On the other hand, our system predicts tags with an accuracy of 90.6% for the 24K test data without any expert knowledge of open-class words. This result is achieved with only a minimal lexicon of closed-class items (about 0.6% of the full lexicon), and is not far from the best previous performance of 93.4% achieved with a full lexicon (LDA+AC with $d = 1$). One other work that investigates reduced lexicons is (Haghighi and Klein, 2006), which develops a prototype-drive approach to propagate categorical properties using distributional similarity features. Using only three exemplars of each tag in the full Treebank tagset, they achieve a tagging accuracy of 80.5% using a larger dataset.

Unsupervised POS tagging with a full dictionary

Even though our system is primarily proposed for reducing the information available to unsupervised tagging, it also works well with the knowledge of a full lexicon. According to a full lexicon, there may be more than one possible tags falling in the same *N/V* category for a word. However, disambiguating between tags in the same *N/V* category is beyond the ability of our disambiguation model. Thus during tagging, when more than one possible tags fall in the predicted N/V category according to a full dictionary, we simply make a

size	12K	24k	48k	96K	#tag	lex.
BHMM2	85.8	84.4	85.7	85.8	17	full
CRF/CE	86.2	88.6	88.4	89.4	17	full
our model	**91.0**	**91.6**	**91.6**	**91.5**	17	full
our model	*93.1*	*93.6*	*93.5*	*93.4*	27	full
our model	88.9	89.3	90.2	90.4	17	closed
our model	*90.9*	*91.2*	*92.0*	*92.2*	27	closed

Table 2.7: Tagging Accuracy of models trained over dataset varying in sizes.

random choice. Although not as constrained as the acquired lexicon, the use of a full lexicon does help improve tagging performance, since the acquired lexicons are still far from perfect. As shown in Table 2.7, our system learns very fast with a full lexicon, but when only a closed-class lexicon is provided, more training data does help improve tagging accuracy, the same learning pattern as other models.

Error Analysis

There are certainly contexts where no functional elements can help with tagging, in such cases our system simply leaves it to chance. Furthermore, if the gold prediction is not listed as a possible tag for a word in the acquired lexicon, then no disambiguation model can correct these errors due to imperfect lexicons. We show in Table 2.8 the number of errors made by the disambiguation model for open-class, by the disambiguations model for closed-class, by random choices and those due to imperfect lexicon. Moreover, in Table

	system with closed-class lexicon		system with full lexicon	
sub-model	#errors	accuracy	#errors	accuracy
open-class	1089	87.3%	3546	78.9%
closed-class	1694	89.6%	1709	89.7%
random	1148	44.2%	981	44.9%
recall	3650	-	75	-
total	7581	75.2%	6311	82.1%
#ambiguous	30563		35229	

Table 2.8: The number of errors and percent *ambiguous* tokens tagged correctly in the 96k dataset with 27 tags. Note that the statistics are over ambiguous tokens only.

2.8, we show the disambiguation accuracy of ambiguous words only by each model as well.

2.8 Totally unsupervised POS tagging

Finally, to achieve a totally unsupervised tagging system with no input of any form of lexicon, we plug the acquired lexicon of closed-class words (Section 2.4), into the two-stage unsupervised tagging system (Section 2.7). Since we still lack the information on possible tags for closed-class words, tagging is evaluated for open-class words only, i.e. for all words that are not in the acquired closed-class lexicon. Tagging accuracy is calculated by the percentage of correctly tagged words out of all words that are not in the acquired closed-

algorithm	measurement	tagging acc.	totally tagged
baseline	$tokenC$	81.97%	604287
	$typeC_{following}$	82.79%	618975
	$typeC_{preceding}$	81.64%	624964
bootstrap	$tokenC$	83.06%	631610
	$typeC_{following}$	**86.54%**	618005
	$typeC_{preceding}$	72.80%	749482
gold closed-class lexicion		*91.54%*	*611028*

Table 2.9: The percentage of correctly tagged tokens out of all predictions. The system tags open-class words only and distinguishes 6 POS categories.

class lexicon. So as to compare with recent work on totally unsupervised POS tagging, we experiment over the whole WSJ corpus. The total number of tagging predictions may vary according to acquired closed-class lexicons by different models.

Since the key theme of the whole chapter is about the distinction between closed- vs. open-class words, whether the proposed algorithm works well for acquiring closed-class words is our main interest. Thus, in Table 2.9, we vary models for acquiring closed-class words in the totally unsupervised tagging experiments, and compare these results to the system with a gold closed-class lexicon. Recall that, as discussed in Section 2.4, for the baseline model we have three alternative diversity measurements to choose from, but for the proposed bootstrapping algorithm, there are four alternatives. For the sake of comparison, we only consider one token-based diversity, $tokenC_{following}$ written as $tokenC$ in Table 2.9,

and ignore the other token-based measurement that takes the preceding word as context. As shown in Table 2.9, the best performance is achieved with the closed-class lexicon acquired by the proposed bootstrapping algorithm, using a type-based diversity measurement which considers the following word as context. Similar with the results reported in Figure 2.1, the baseline model is not that sensitive to the choice of diversity measurement, however, the bootstrapping algorithm works well only with a carefully designed measurement function.

In recent work on totally unsupervised POS tagging, the state-of-art tagging accuracy is reported by Blunsom and Cohn (2011) as 77.5% with a many-to-one mapping of induced clusters to gold POS tags, and comparable results are also reported in (Teichert and Daume, 2010; Abend et al., 2010; Graca et al., 2009; Moon et al., 2010). Such a many-to-one mapping maps each induced cluster to the gold tag that is preferred by most words in that cluster, thus frequency information that are obtained from annotated corpora is unfairly used. If each word forms a cluster by itself, then this many-to-one mapping actually assigns the most frequent tag to each word type, a well-known good baseline model to the supervised POS tagging problem. By a more constrained mapping of the induced clusters, the state-of-art performance is reported by (Lamar et al., 2010) as 59.3%. Compared to these results, the tagging performance reported here, 86.5% for open-class words of six categories, is quite promising. Since these works use finer tags than we do here, we may appear to be reporting on a simpler task. However, it has proven difficult for these systems to use further agglomerative processing to induce simple distinct syntactic categories which map to POS tags naturally (Christodoulopoulos et al., 2010). Therefore, achieving high accuracy with a smaller tag set is the harder, not easier, task for those systems.

2.9 Conclusion

We propose a bootstrapping algorithm acquiring functional elements in both free and bound forms. After applying this algorithm to acquire closed-class words and morphological endings in different experiments, we use the acquisition outputs to build a totally unsupervised POS tagging system. Without any lexicon input, the totally unsupervised POS tagging system tags 6 lexical categories with a rather promising performance, and with the input of a closed-class lexicon, which contains only about 0.6% word types of the full lexicon, the proposed two-stage unsupervised POS tagging system can achieve a tagging accuracy comparable to the so-called unsupervised models that requires the input of full lexicons.

Even though we have emphasized the theme of this chapter as capturing the distinction between closed- and open-class words, the deeper motivation of this series of work is actually the feature-based analysis of part-of-speech tags. Given our feature-based view of POS tags, the tagging problem is not necessarily formulated as a sequence labeling problem. Instead, we argue that, if we only concern with coarse lexical tags, except for one binary syntactic feature that needs to be disambiguated during tagging, all other features of POS tags, such as semantic and morphological features, can be settled by global learning processes (as opposed to locally disambiguation). For example, semantic features can be set by POS induction and morphological features can be set by morphology learning.

The distinction between closed- and open-class words is then crucial in practicing such a feature-based view for unsupervised POS tagging. First of all, the decomposing story is only for lexical categories, i.e. open-class words, in contrast, closed-class words are much

more ad-hoc but enumerable. Second, in a distributional clustering for POS induction, we only use functional elements in the local context of a word to represent it in the feature space. Third, so as to make a better use of unambiguous cases to establish disambiguation rules, the tagging system is designed to learn and tag open-class words first and dealing with closed-class words afterwards. Finally, the disambiguation rules for open-class words are conditioned on closed-class words in their local contexts. Moreover, since we first highlighted the distinction between closed- and open-class words in (Zhao and Marcus, 2009), the idea of distinguishing closed-class words from other words has achieved more and more attention in following works on unsupervised POS tagging by others, such as (Graca et al., 2009; Teichert and Daume, 2010; Moon et al., 2010) etc.

Neither a feature-based view of lexical categories nor the distinction between closed- and open-class words is new to linguists, but both of them have been neglected for engineering use. By exploring these linguistic aspects, we decompose the POS tagging problem to subproblems of (globally) POS induction and (locally) binary feature disambiguation, thus reduce the resource required for learning an overall tagging model. All recent research on tagging, either supervised or unsupervised work, views POS tagging as a sequence labeling problem and treats all POS tags as equivalently meaningless labels. We are not aware of any other effort on exploring the POS tagging problem with a new formulation. Given the successful application of statistical models in supervised POS tagging, one may wonder why we are interested in exploring new formulations of POS tagging, but noticing that how severe the lack of supervision affects the performance of unsupervised POS tagging, we do need to open our mind to new angles of the POS tagging problem.

Chapter 3

Long-tail Distributions and Morphology Learning

3.1 Introduction

A [4] morphological analyzer takes words as input and gives morphological analysis to each word. For example, suppose we describe morphological structures of English words with a segmentation-based model, then for word *giving*, the output analysis is composed of a stem *giv-* and a suffix *-ing*. Most work on morphology learning is unsupervised, thus, in this work, when we talk about morphology learning, it is implicitly assumed to be unsupervised. Unsupervised morphology learning acquires morphological analyzers from raw text only.

We are aware of descriptions of word structures in other levels than just the segment-based model. For example, one may be only interested in deciding whether two words

[4] This paper extends (Zhao and Marcus, 2012b), which is basically Section 3.7.

are inflections or derivations of the same word stem, then we do not need a analyzer that clearly draws a boundary between their stems and suffixes. Or, one may be interested in describing word structures as a result of transformations. For example, for word *giving*, a morphological rule *e→ing* can be inferred, which tells that *giving* is an inflectional form of *give* and describes how the bare form is inflected according to the transformation rule.

Moreover, when we generally talk about word structures, we are not making distinctions between inflections, derivations, or word formations. When a word undergoes an inflection, only the associated syntactic category is changed, i.e. the word is expected to occur in a new syntactic context, but preserves the same lexical meaning. For example, when the '*-ing*' inflectional rule applies, *giving*, the present progressive form of *give*, no longer functions as a (non-)finite verb. In contrast, if we add a prefix *be-* to word *give*, then a new word *begive*, taking a new lexical meaning, is composed by derivation. Moreover, when a new word is composed of two or more words, such as *laptop* or *nevertheless*, it is usually considered as word formation, but less as undergoing a morphological transformation.

In this work, we will stick with the segmentation model of morphology, so as to compare with previous work of interest. Following (Goldwater et al., 2006), we will also focus our experiments on inflectional morphology, and try various approaches to acquire morphological models that segment verbs into stems and suffixes. This chapter organizes as:

- Random guesses and the experimental setup: Section 3.2.
- A rule-based approach given acquired suffixes: Section 3.3.
- Learning with the Expectation Maximization (EM) algorithm: Section 3.4.

- Gibbs sampling from multinomial distributions: Section 3.5.
- Gibbs sampling from log-normal distributions: Section 3.7.

3.2 The first guesses and experimental setup

So as to set up the experiment framework for other approaches, let us first see how simple guesses work for this problem. The first thought is to segment each word randomly. For example, considering word *giving* again, we may randomly pick one out of the 7 possible analyses with equally likely chances. Moreover, if we know that, in English, it is suffixes but prefixes of words that reflect inflections, and word stems usually contain at least 3 characters, then we can reduce the number of possible segmentations of *giving* from 7 to 4.

Instead of the random guess, we can also simply leave all the words alone, i.e. output the word itself as the stem together with an empty suffix. With this rather conservative guess, all the verbs with no inflections are guaranteed to receive the correct analyses, but all the inflected forms of verbs will always receive wrong analyses.

3.2.1 Evaluation criteria

It is hard to tell which one of the above guesses leads to better morphological analyses. We will consider two criteria to evaluate the prediction accuracy:

- Type-based accuracy: the percentage of correctly analyzed word types out of all distinct word types in the data.

- Token-based accuracy: the percentage of correctly analyzed tokens (word occurrences) out of all tokens in the data.

Even though type-based evaluations have been used more in previous work, there are at least two advantages of the token-based criterion. With a random guess, there may be different analyses given to the different occurrences of the same word type in the input data. This randomness can be captured by a token-based evaluation but not a type-based evaluation, since with a type-based evaluation, only one analysis is considered for each word type. Secondly, a token-based criterion gives more weights to more frequent words, thus respects more of the real distribution of unprocessed text data.

3.2.2 Prepare gold data

Following (Goldwater et al., 2006), we will evaluate inflectional analyses of all the verbs in the WSJ corpus. With respect to the Penn Treebank guideline (Marcus et al., 1993), we consider words that are associated with POS tags of 'VB', 'VBP', 'VBZ', 'VBD', 'VBN' or 'VBG' as verbs. Then using the gold POS annotations, we extracted 137,899 verbs from the whole WSJ corpus which belong to 7,708 distinct word types. With heuristics based on POS tags and spellings, we automatically segment each verb into a stem, which cannot be empty, and an inflectional suffix, which may be empty, and use these segmentations as gold standards to evaluate our experiments on morphology learning.

More specifically, to automatically segment each verb, for a verb tagged as 'VBZ', if it ends with *-s* then divide the word at *-s*, except for *goes*, *does* and words ending in *-xes* or

	type-based acc.	token-based acc.
random guess with all possibilities	12.27%	18.39 %
random guess with constrained possibilities	22.11%	46.71%
conservative guess	30.53%	56.99%

Table 3.1: Evaluate the first guesses with both type-based and token-based criteria.

-ches which are divided at *-es*. For a verb tagged as 'VBD' or 'VBN', if it ends with *-ed* then divide the word at *-ed*; or if it is an irregular verb and ends with *-en* or *-n*, then divide the verb at *-en* or *-n* respectively. [5] By default, each word has an empty suffix.

3.2.3 Preliminary Experiments

So as to get a preliminary sense of inflectional analyses of the verbs, we compare the conservative guess with the random guess. Moreover, for the random guess, we experiment with two implementations of this idea: 1) randomly picks one segmentation out of all possible segmentations including those of empty prefix; and 2) consider only the possible segmentations that consist of prefixes containing at least 3 characters.

As shown in Table 3.2.3, none of these first thoughts works well, but we can still read interesting patterns from the comparison. Firstly, we have discussed that, with random guesses, different occurrences of the same word type may receive different analyses. Thus the difference between the random guess and the conservative guess is more distinctively shown with a token-based evaluation than a type-based evaluation. Secondly, since a

[5] We use the list of irregular verbs provided by `http://www.englishpage.com`

token-based evaluation gives more weights to more frequent words, the improvement introduced by the conservative guess is much better realized with a token-based evaluation. This pattern complies with our knowledge of English that irregular verbs, such as *ran* and *said*, most of which have an empty suffix according to our gold data, tend to be of high frequency. Finally, when we constrain the possible segmentations for random guesses, the resulting improvement is also more distinctively shown with a token-based evaluation. Overall, even though type-based evaluations are more popular in literature, we have shown that a token-based evaluation does help distinguish a better model.

3.3 A rule-based approach

If lists of 'justified' stems and suffixes are given, we can divide each word by a very simple rule: a legal segmentation must be composed of a justified stem and a justified suffix. More formally, given a set of stems $\mathbb{T}$ and a set of suffixes $\mathbb{F}$, we can divide a word w into stem t and suffix f, only if $t \in \mathbb{T}$, $f \in \mathbb{F}$ and $w = t.f$. For example, if $\mathbb{T} = \{$*'laugh-'*, *'analyz-'*$\}$, and $\mathbb{F} = \{$*'-ed'*, *'-s'*$\}$, then *'analyzed'* can be segmented into *'analyz-'* and *'-ed'*, but *'red'* won't be segmented. Thus, with this rule-based approach, morphology learning can be considered acquiring sets of justified stems and suffixes.

When the acquired set of stems or suffixes is imperfect, by this simple segmentation rule, it is certainly possible to give a word more than one legal analyses. For example, if $\mathbb{T} = \{$*'laugh-'*, *'analyz-'*, *'analyze-'*$\}$, and $\mathbb{F} = \{$*'-ed'*, *'-s'*, *'-d'*$\}$, where ending *'-d'* is wronly acquired as a justified suffix, then the proposed segmentation rule allows for two

possible segmentations: *analyz-ed* or *analyze-d*. In practice, our analyzer produces the segmentation that contains the suffix with a higher diversity. However, with perfect sets of justified stems and suffixes, we are not aware any English word that is ambiguous regarding this segmentation model. On the other hand, in theory, a word may be composed of a justified stem and a justified suffix, but the stem and suffix do not form a proper morphological segmentation. However, for English verbs, we are not aware of such particular examples.

Recall that, in Section 2.5, we have already dealt with the acquisition of morphological suffixes. The proposed bootstrapping algorithm is originally designed for the acquisition of closed-class words, but perfectly applicable to morphology learning. In addition to a list of suffixes, the algorithm also generates a complementary set of proper contexts, i.e. a set of justified stems ready for our use in the case of morphology learning. We are going to elaborate the bootstrapping algorithm as a complete story for morphology learning below.

3.3.1 A bootstrapping algorithm acquiring morphological units

In this section, we are going to describe a bootstrapping algorithm for acquiring morphological units, especially stems and suffixes for English. In Section 2.4, the proposed bootstrapping algorithm was originally developed for the acquisition of closed-class words, and in Section 2.5, we introduced how it is applied to morphology learning. Briefly speaking, the bootstrapping algorithm is designed based on the following two main ideas:

- **Diversity measurement**: functional elements, including morphological units and functional words, tend to occur in more diverse contexts.

- **Proper contexts**: the computation of diversity for each item should be carried out according to properly justified contexts only.

The algorithm iteratively generates a set of functional elements and a set of justified contexts, and for either set, diversity measurements of its items are computed according to the other set. As the bootstrapping proceeds, either set becomes more and more accurate and so the diversity measurement of the items in the other set. For both the acquisition of closed-class words and morphological units, we experiment with two kinds of diversity measurement: type-based or token-based. With the former counting the distinct types of justified contexts an item ever occurs in, and the latter counting the total occurrences of an item occurring in justified contexts. The main difference between the two applications of the same bootstrapping algorithm is that for acquiring closed-class words, we need to consider either the preceding word or the following word as the context of a word in sequence; however, for acquiring suffixes, we only consider the corresponding stem as the context of the suffix given any division of a legal word. Therefore the filtering step that keeps the two generated sets as complementary as possible is no longer required for morphology learning, even though no harm will be done with this redundant step.

More formally, given a corpus $\mathbb{C}$ and the set of all word types in the corpus $\mathbb{W}$. Regarding a set of acquired stems $\mathbb{T}$, we measure the contextual diversity of a suffix f with a measurement function div, which can be either type-based or token-based as discussed

above, i.e.

$$div(f, \mathbb{T}) = \begin{cases} \sum_{w \in \mathbb{W}} \mathbb{1}(w = t.f) & \text{for type-based measurement} \\ \sum_{w \in \mathbb{C}} \mathbb{1}(w = t.f) & \text{for token-based measurement} \end{cases}$$

$$\mathbb{1}(w = t.f) = \begin{cases} 1 & \text{if } t \in \mathbb{T} \\ 0 & \text{if } t \notin \mathbb{T} \end{cases}$$

Similarly, regarding a set of acquired suffixes $\mathbb{F}$, the diversity of a stem t is measured as

$$div(t, \mathbb{F}) = \begin{cases} \sum_{w \in \mathbb{W}} \mathbb{1}(w = t.f) & \text{type-based} \\ \sum_{w \in \mathbb{C}} \mathbb{1}(w = t.f) & \text{token-based} \end{cases} \qquad \mathbb{1}(w = t.f) = \begin{cases} 1 & \text{if } f \in \mathbb{F} \\ 0 & \text{if } f \notin \mathbb{F} \end{cases}$$

According to our assumption, when $div(t, \mathbb{F}) > 1$, t can be justified as a proper context.

Algorithm 2 The algorithm for acquiring morphological stems and suffixes

Require: A corpus $\mathbb{C}$ containing raw text only.

Require: A total number of acquired suffixes K.

Initialize set $\mathbb{F}_0$ to be empty.

Initialize set $\mathbb{T}_0$ to contain all possible suffixes.

for $k = 1...K$ **do**

Let $\mathbb{F}_k$ contain k suffixes with the highest diversities measured by $div(f, \mathbb{T}_{k-1})$.

Let $\mathbb{T}_k$ contain such stems that $div(t, \mathbb{F}_k) > min(k - 1, 1)$.

end for

return $\mathbb{F}_K$ and $\mathbb{T}_K$

We repeat Algorithm 1 as Algorithm 2 here with different notations, so as to make a coherent story with other approaches in this chapter. The corpus $\mathbb{C}$ required by the algorithm contains raw text only without any form of annotation. Besides, we also need a total number of iterations K to control the stop condition, i.e. to specify how many suffixes we wan to acquire from the input corpus. Set $\mathbb{F}_0$ is initialized to be empty and set $\mathbb{T}_0$ may be simply initialized to contain all possible suffixes of all words in the corpus $\mathbb{C}$. At the k_{th} bootstrapping iteration, $k > 0$, we compute set $\mathbb{F}_k$ as the top k suffixes of the highest contextual diversity according to set $\mathbb{T}_{k-1}$. Accordingly, at the k_{th} bootstrapping iteration, $k > 1$, we compute set $\mathbb{T}_k$ of such stems that can form legal words with more than one acquired suffixes in $\mathbb{F}_k$. Since the diversity measurement of suffixes varies over iterations with respect to the updated set of stems, the ordering of the acquired suffixes may also vary.

Acquisition output

In Section 2.5, we have shown the acquisition output by the bootstrapping algorithm over the whole WSJ corpus, however, in this chapter, we focus on the learning result over verbs only in the WSJ corpus. Moreover, so as to compare with other models more fairly, we are not implementing special mechanisms for removing complex suffixes such as *-ers*, *-ions* or *-ings*, neither the trick for removing the most noisy suffix *-e* (Zhao and Marcus, 2011). Even without these special treatments, as shown in Table 3.2, most acquired suffixes make sense as morphological units.

To show more examples of acquisition output, we can alternate the choice of diversity

kth iter.	set $\mathbb{F}_k$	size of set $\mathbb{T}_k$
1st	*'-d'*	0
2nd	*'-d', '-s'*	2481
3rd	*'-s', '-d', '-ing'*	407
4th	*'-s', '-d', '-ing', '-ed'*	806
5th	*'-ing', '-ed', '-s', '-d', '-e'*	1824
6th	*'-ed', '-ing', '-s', '-e', '-d', '-es'*	2068
7th	*'-ed', '-ing', '-s', '-e', '-es', '-d', '-en'*	2155
8th	*'-ed', '-ing', '-s', '-e', '-es', '-d', '-en', '-t'*	2165
9th	*'-ed', '-ing', '-s', '-e', '-es', '-d', '-t', '-en', '-n'*	2185
10th	*'-ed', '-ing', '-s', '-e', '-es', '-d', '-n', '-t', '-en', '-ned'*	2210

Table 3.2: The acquisition output by Algorithm 2 with type-based diversity measurement over all verbs in the WSJ corpus.

measurement as well as the choice of initialization condition. And the baseline model we compare with simply ranks the suffixes according to a fixed set of stems. As shown in Table 3.3, by either the bootstrapping algorithm or baseline model, learning with the type-based measurement performs better. Moreover, both the learning over the whole WSJ corpus (Table 2.3) and the learning over the verbs only (Table 3.3) shows the same preference of model. With experiments over different corpora, the proposed bootstrapping algorithm with the type-based measurement is shown to be the best model for morphological learning,

alg.	div.	init.	output at the 10th iteration
baseline	token	$\mathbb{B}^1$	'*-d*', '*-ed*', '*-s*', '*-e*', '*-g*', '*-ng*', '*-ing*', '*-as*', '*-id*', '*-re*'
		$\mathbb{B}^2$	'*-ed*', '*-d*', '*-s*', '*-ing*', '*-aid*', '*-ays*', '*-re*', '*-en*', '*-n*', '*-ay*'
	type	$\mathbb{B}^1$	'*-d*', '*-ed*', '*-g*', '*-ng*', '*-ing*', '*-s*', '*-e*', '*-ted*', '*-es*', '*-ting*'
		$\mathbb{B}^2$	'*-s*', '*-ing*', '*-ed*', '*-d*', '*-es*', '*-n*', '*-ting*', '*-ping*', '*-ped*', '*-ted*'
bootstrap	token	$\mathbb{B}^1$	'*-d*', '*-s*', '*-re*', '*-id*', '*-ve*', '*-ys*', '*-t*', '*-y*', '*-es*', '*-ing*'
		$\mathbb{B}^2$	'*-ed*', '*-s*', '*-d*', '*-ing*', '*-e*', '*-as*', '*-ve*', '*-es*', '*-ere*', '*-t*'
	type	$\mathbb{B}^1$	'*-ed*', '*-ing*', '*-s*', '*-e*', '*-es*', '*-d*', '*-n*', '*-t*', '*-en*', '*-ned*'
		$\mathbb{B}^2$	'*-ed*', '*-ing*', '*-s*', '*-e*', '*-es*', '*-d*', '*-n*', '*-t*', '*-en*', '*-ned*'

Table 3.3: The acquisition output learnt from all verbs only in WSJ corpus. Set $\mathbb{B}^1$ contains all possible suffixes and $\mathbb{B}^2$ contains legal words as proper stems only.

and this bootstrapping model performs stably with different initializations.

3.3.2 Experiments

Given acquired stems and suffixes, we can build a simple rule-based model to segment words. Besides the more conservative strategy discussed at the beginning of this section, we also have a more greedy strategy as an alternative to build the segmentation model.

- **Greedy strategy**: given a set of acquired suffixes $\mathbb{F}$ only, we can divide a word w

into $t + f$, as long as $w = t.f$ and $f \in \mathbb{F}$.

- **Conservative strategy**: given both a set of acquired suffixes $\mathbb{F}$ and a set of acquired stems $\mathbb{T}$, we may divide a word w into $t + f$, only if $w = t.f$, $f \in \mathbb{F}$ and $t \in \mathbb{T}$.

Without the set of justified stems, the segmentation model applies in a more greedy way and may cause more false positive errors due to the over-segmentation of irregular verbs. On the other hand, constrained by a set of acquired stems as well as a set of acquired suffixes, we may be too conservative and miss segmentations of many regular verbs. As shown in Table 3.4, the rule-based model with a more conservative strategy performs better with the toke-based evaluation, but the one with a more greedy strategy performs better with the type-based evaluation. This pattern is consistent with our previous experiments on the random guess and the conservative guess. Given that irregular verbs in English tend to be of high frequency, a more conservative segmentation model, which generates less false positive errors by avoiding the over-segmentation of irregular verbs, performs better with the token-based evaluation.On the other hand, regular verbs form the majority of the verb types, thus a greedy application of the rules, which aggressively predicts more segmentations, performs better with the type-based evaluation. For a more straightforward look into the errors, confusion matrices are shown in Figure 3.1 and Figure 3.2.

We have used the acquisition output by the bootstrapping algorithm with type-based diversity measurement to build rule-based segmentation models. As shown in Table 3.2, all inflected endings are acquired at the 10th iteration, as well as noisy outputs *'-e'* and *'-d'*.

	type-based acc.	token-based acc.
random guess with all possibilities	12.27%	18.39 %
random guess with constrained possibilities	22.11%	46.71%
conservative guess	30.53%	56.99%
rule-based with acquired stems (conservative)	69.75%	**73.98**%
rule-based without acquired stems (greedy)	**79.01**%	49.55%

Table 3.4: Evaluate rule-based models with both type-based and token-based criteria.

3.3.3 Randomness and Learning

Besides the random guess, both the conservative guess and the rule-based models give a certain morphological analysis to all occurrences of the same word type. Even though it complies with our common sense to assign one certain analysis to a word type, the randomness allowed in a morphological model is crucial for probabilistic learning. In the contrast of the proposed bootstrapping algorithm, which requires a special stop condition, a successful probabilistic learning process is expected to converge to a stable state. For example, a Markov Chain Monte Carlo (MCMC) method constructs a Markov chain with the desired distribution as its equilibrium distribution. In Section 3.4 and Section 3.5, we assume multinomial distributions of segmentations, and learn with the Expectation Maximization algorithm (EM) and a MCMC method, Gibbs sampling, respectively. In Section 3.7, we assume log-normal distributions instead and use Gibbs sampling for learning.

When randomness is allowed, besides type-based input that has been assumed so far , a probabilistic learning may also observe token-based input. In the contrast of the type-

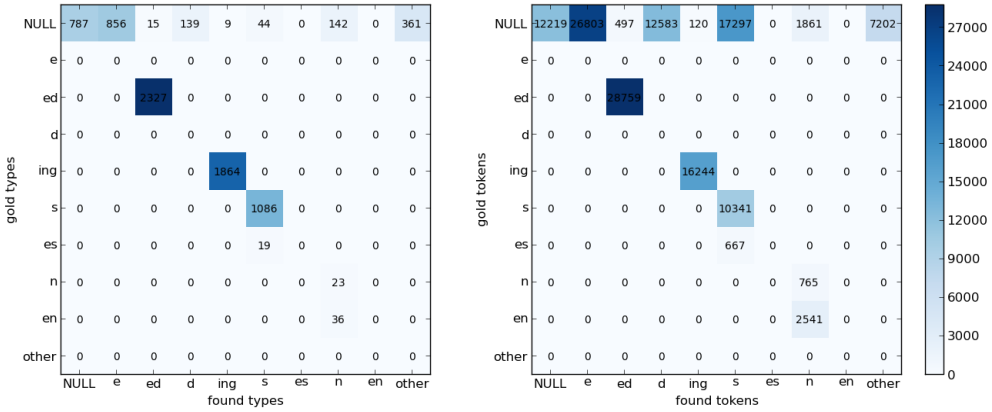


Figure 3.1: The confusion matrices for the greedy rule-based model.

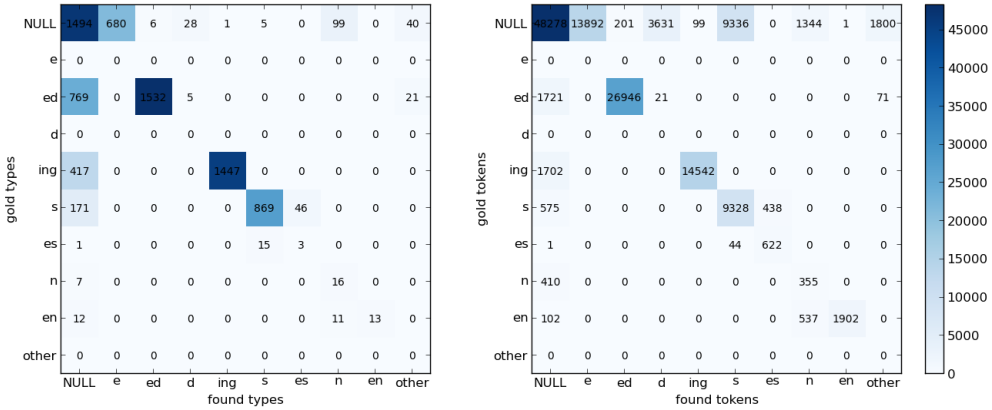


Figure 3.2: The confusion matrices for the conservative rule-based model.

based input, in which each word type occurs only once, token-based input reflects the real distribution of word frequency. It may be worth pointing out that there is no according relation between the two forms of input and the two evaluation criteria. In other words, a morphology model trained over token-based input can also be evaluated with the type-based criterion, or vice versa (with four variations in total).

3.4 EM learning

It is to our surprise that, the Expectation Maximization (EM) algorithm, which is extensively used for unsupervised learning, is not applied for morphology learning as widely as one may expect. Instead, Bayesian inference approaches seem much more popular on this topic, e.g. a maximum a posteriori (MAP) probability estimate in (Goldsmith, 2001) and (Creutz and Lagus, 2007) and Gibbs sampling in (Goldwater et al., 2006), (Lee et al., 2011), (Moon et al., 2009) and etc. However, as we are going to show, EM is, again, a baseline model worth exploration, which performs very well for this task considering its simple implementation. Moreover, since we are usually more familiar with EM then Bayesian approaches, applying EM first help understand the MCMC method (Gibbs sampling) we are going to introduce in the next section. After all, EM can be viewed as a forerunner of MCMC methods in that its data augmentation step replaces simulation in MCMC methods by maximization.

3.4.1 EM for multinomial

In morphology learning, we aim to learn a joint distribution of segmentations and words $P(\mathbf{S}, \mathbf{W}|\theta)$, but only observe unprocessed words $P(\mathbf{W})$. In other words, both segmentations, $\mathbf{S}$, and parameters θ are hidden. Therefore, we may resort to the EM algorithm that enables parameter estimation with incomplete data. Each iteration of the EM algorithm consists of an expectation step (E-step) followed by a maximization step (M-step). E-step estimates the sufficient statistics of the complete data given the observed data. Then M-

step takes the estimated complete data and estimates θ by maximum likelihood (MLE) as though the estimated data were the observed data (Dempster et al., 1977).

In our case, denote π_t the probability of stem t given parameters π, i.e. $\pi_t = P(t|\pi)$; and ϕ_f the probability of suffix f given parameters ϕ, i.e. $\phi_f = P(f|\phi)$. Assume that stems and suffixes are independently distributed, then we have

$$P(s = (t, f)|\pi, \phi) = P(t|\pi)|P(f|\phi) = \pi_t \times \phi_f,$$

where segment $s = (t, f)$ indicates that a word is composed of stem t and suffix f. Denote the number of stem t as N_t and number of suffix f as N_f in the estimated complete data. Assume that both suffixes and stems are multinomially distributed, then, at each M-step, MLE estimates the current parameters π and ϕ by exact solutions as follows:

$$\hat{\pi_t} = P(t|\hat{\pi}) = \frac{\mathrm{E}[N_t]}{\sum_t \mathrm{E}[N_t]} \quad \text{for each stem } t, \text{ and}$$

$$\hat{\phi_f} = P(f|\hat{\phi}) = \frac{\mathrm{E}[N_f]}{\sum_f \mathrm{E}[N_f]} \quad \text{for each suffix } f.$$

Then at each E-step, we compute sufficient statistics of the estimated complete data using the current parameters $\hat{\phi}$ and $\hat{\pi}$. The expected value of N_t is calculated as

$$\begin{aligned}\mathrm{E}[N_t] &= \sum_{\text{suffix } f} P(s = (t, f)|\hat{\pi}, \hat{\phi}) \times n_{(w=s)} \\ &= \sum_{\text{suffix } f} \hat{\pi_t} \times \hat{\phi_f} \times n_{(w=(t,f))} \quad \text{given our independence assumption.}\end{aligned}$$

where denote as $n_{(w=s)}$ the number of observed words that can be segmented as s or when $s = (t, f)$, $n_{(w=(t,f))}$ denotes the name number. Similarly, the expected value of N_f is

calculated as follows:

$$\mathrm{E}[N_f] = \sum_{\text{stem } t} \hat{\pi}_t \times \hat{\phi}_f \times n_{(w=(t,f))}.$$

3.4.2 Experiments

Following the experiment framework described in Section 3.2.3, we also evaluate EM learning with both the type-based and token-based criteria. The learning curves are shown in Figure 3.3 and 3.4, for type-based and token-based evaluation respectively.

Type-based vs. Token-based input. As discussed in Section 3.3.3, besides the type-based input, which contains distinct word types only, we may also run EM over token-based input, which contains the raw text data that reflects the real distribution of word frequency. As shown with both the type-based and token-based evaluation, the from of the input data to EM does matter. For both evaluations, EM learns well with type-based input and converges to a stable distribution; however, given the token-based input, the EM algorithm is not able to jump out of a local maximum, for which we are not sure how to help.

Constraints on possible segmentations. At each expectation step, the EM algorithm computes probabilities for each possible completion of the missing data, instead of picking the single most likely completion. In our case, when we compute the expected value of the count of stems or suffixes, all possible segmentations of each word are weighted in. Given a list of suffixes, e.g. the acquisition output of the proposed bootstrapping algorithm in Section 3.3.1, we may allow only the divisions of the word that produces the suffixes in the given list as legal segmentations. Constraining the possible completions of data does speed

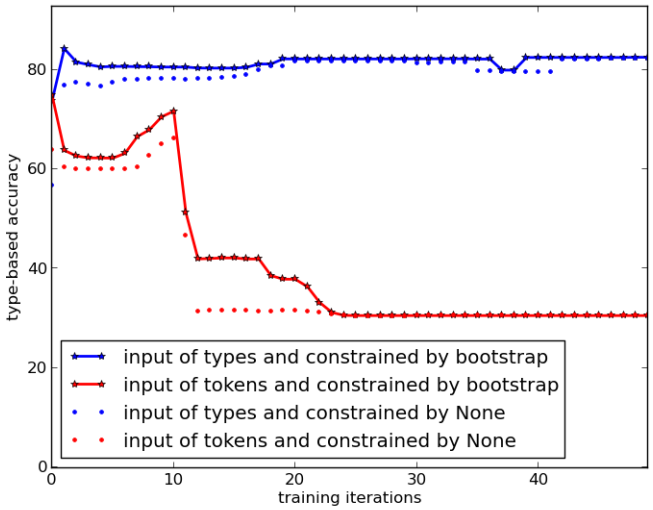


Figure 3.3: The type-based accuracy of EM.

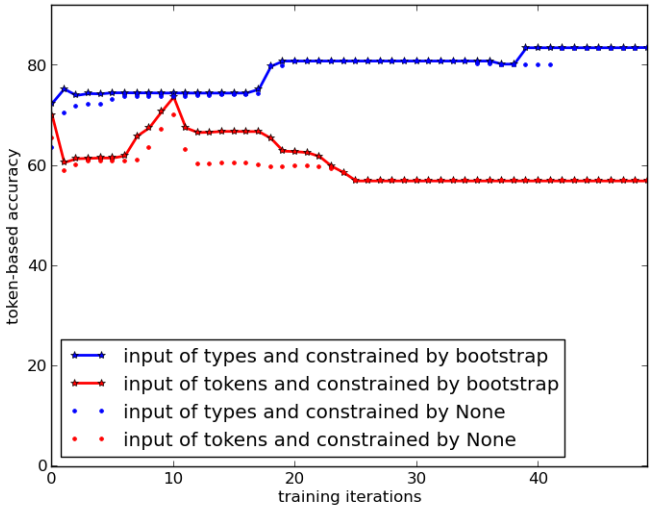


Figure 3.4: The token-based accuracy of EM.

convergence, as shown in both Figure 3.3 and 3.4, however, the level of performance is not affected with either type-based or token-based evaluation.

As shown in Table 3.5, EM outperforms the rule-based models with both the type-based

and token-based evaluation. We show confusion matrices for EM in Figure 3.5, and for a easier comparison, we repeat Figure 3.1 as Figure 3.6, which depicts confusion matrices for the rule-based model with the greedy strategy. As shown in these confusion matrices, EM makes much less mistakes on verbs that are not inflected [6].

	type-based acc.	token-based acc.
rule-based with acquired stems (conservative)	69.75%	73.98%
rule-based without acquired stems (greedy)	79.01%	49.55%
EM with type-based input and not constrained	**82.19**%	**83.50**%

Table 3.5: Evaluate EM with both type-based and token-based criteria.

3.5 Gibbs sampling from multinomial distributions

In the last section, for morphology learning, we use the EM algorithm to learn from incomplete data . The EM algorithm iteratively proceeds with the following two steps:

- computes sufficient statistics of the complete data given the current parameters.
- estimates parameters by maximization as though the estimated data were observed.

By using the EM algorithm to find maximum likelihood estimates of π and ϕ, we maximizes the likelihood of observed words given the segmentation model, i.e. $P(W|\pi, \phi)$. As

[6] In (Zhao and Marcus, 2012b), a higher type-based accuracy (83.59%) is reported for the greedy rule-based model, trained over the whole WSJ corpus by the bootstrapping algorithm.

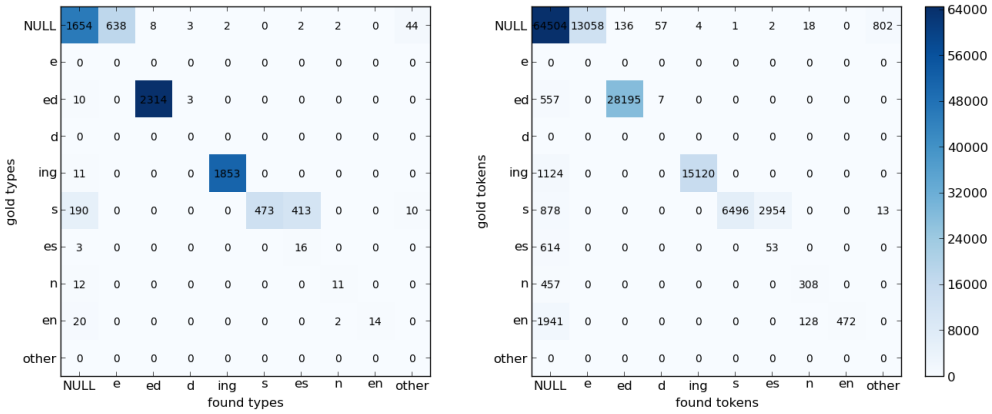


Figure 3.5: The confusion matrices for EM.

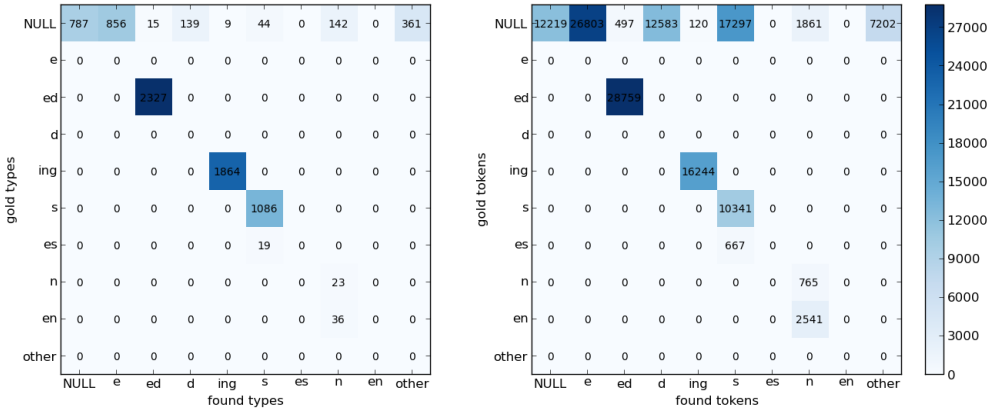


Figure 3.6: The confusion matrices for the greedy rule-based model.

we have seen in Figure 3.3 and 3.4, using EM may encounter the problem of local maximums. Thus we can also try Markov Chain Monte Carlo (MCMC) methods, which relate to EM in that its data augmentation step replaces maximization in EM by simulation.

Suppose that we are able to generate random draws from the target distribution, θ, of segmentations, corresponding to i.i.d. random variables. The obvious estimate of $\mathrm{E}[\theta_s]$ is

the empirical average:

$$\hat{\theta}_s = \frac{1}{m} \sum_{t=1}^{m} \theta_s^{(t)},$$

where $\theta_s^{(t)}$ can be directly computed with the complete data observed in the t_{th} draw from the target distribution. This is the ordinary Monte Carlo estimation. However, to draw a perfect random sample from a target distribution is usually impractical. Instead, while using Markov Chain Monte Carlo (MCMC) methods, we construct a ergodic Markov chain with the limit distribution θ, and draw samples from the Markov chain. Each state of the Markov chain is an assignment of values to the variables being sampled. By Gibbs sampling, the next state is reached by sequentially sampling all variables from their distribution when conditioned on the current values of all other variables and the data. Then with a sufficiently large m, the probability values are independent of the starting values and approaches the stationary distribution.

In this work, we use Gibbs sampling to construct a Markov chain $\mathbf{S}^1, \mathbf{S}^2, \cdots, \mathbf{S}^m, \cdots$, whose stationary distribution is the joint distribution of segmentations and words, $P(\mathbf{S}, \mathbf{W})$. The joint distribution $P(\mathbf{S}, \mathbf{W})$ can be specified by a generative model, which randomly generates data from its posterior distribution. In this section, we are going to review a generative model with multinomial distributions, which is studied in (Goldwater et al., 2006). In Section 3.7, we propose a generative model with log-normal distributions, so as to account for the real distribution of word frequency.

3.5.1 The probability model

Assume that both stems t and suffices f are multinomially-distributed, with parameters π and ϕ respectively, i.e.

$$t_i|\pi, \quad f_i|\phi \sim \text{Multinomial}(\pi), \quad \text{Multinomial}(\phi).$$

This assumption is mainly made for the sake of simplicity in computations, the same reason as for EM. So as to have conjugacy, we assume Dirichlet priors for both stems and suffixes, generated with hyperparameters α and β respectively, i.e.

$$\pi, \quad \phi \sim \text{Dirichlet}(\alpha), \quad \text{Dirichlet}(\beta).$$

Moreover, given our independence assumption,

$$P(s = (t, f)|\pi, \phi) = P(t|\pi)|P(f|\phi) = \pi_t \times \phi_f. \tag{3.5.1}$$

3.5.2 Sampling process

Denote as $s_1...s_N$ the sampled segmentations of words $w_1, ..., w_N$. Assume morphological analyses are independently and identically distributed. Then a weaker assumption directly follows: the finite set of segmentations $\{s_1, ..., s_N\}$ is exchangeable, i.e. $\Pr(s_1, ..., s_N) = \Pr(s_{\Sigma(1)}, ..., s_{\Sigma(N)})$, where Σ is a permutation of the integers from 1 to N. Given the assumption of exchangeability, we can use a simple and widely-used Markov Chain Monte Carlo method, Gibbs sampling, for the inference of generative models.

By Gibbs sampling, we alternatively sample all variables specified in the probability model, with conditional probabilities conditioned on the current values of all other variables

and the data. In our case, denote $\mathbf{S}_{-i}$ the segmentations of all words but the i_{th} word, i.e. $\mathbf{S}_{-i} = \{s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_N\}$. According to the probability model defined above, for each word w_i, in theory, we need to first sample Dirichlet prior parameters conditioned on $\mathbf{S}_{-i}$ and then sample segmentation s_i conditioned on the current parameters.

Furthermore, the parameters π and ϕ can be integrated out, resulting in a more efficient inference procedure. Consider

$$P(f_i = f|\mathbf{S}_{-i}) = \int P(f_i = f|\pi)P(\pi|\mathbf{S}_{-i})d\pi, \tag{3.5.2}$$

where π denotes a multinomial distribution over suffixes and the integral is over all such distributions. By Bayes' rule, we have

$$P(\pi|\mathbf{S}_{-i}) \propto P(\mathbf{S}_{-i}|\pi)P(\pi).$$

Since $P(\pi) \sim$ Dirichlet(α) and conjugate to $P(\mathbf{S}_{-i}|\pi)$, the posterior distribution $P(\pi|\mathbf{S}_{-i})$ will be Dirichlet($\alpha + N_f$), where N_f is the number of segmentations that contain suffix f in $\mathbf{S}_{-i}$. The integration in 3.5.2 is simply the expected value of this posterior Dirichlet distribution, which is calculated as

$$P(f_i = f|\mathbf{S}_{-i}) = \frac{\alpha + N_f}{\alpha_+ + N},$$

where $\alpha_+ = \alpha \times k_F$, if we assume symmetric Dirichlet priors and the number of distinct stem types is k_F. Similarly, ϕ can also be integrated out and we have

$$P(t_i = t|\mathbf{S}_{-i}) = \frac{\beta + N_t}{\beta_+ + N},$$

where $\beta_+ = \beta \times k_T$. Putting together the results, we obtain the conditional probability to sample segmentation s_i for word w_i as follows:

$$P(s_i = s = (t, f)|w_i, \mathbf{S}_{-i}) \propto \mathbb{1}(w_i = t.f)\frac{\alpha + N_t}{\alpha \times k_T + N}\frac{\beta + N_f}{\beta \times k_F + N}.$$

where $\mathbb{1}(w = t.f)$ takes on value 1 if concatenation $t.f$ forms word w, otherwise 0.

Since the parameters are integrated out and do not need to be sampled, we only need to sequentially sample the morphological analysis of each word from its conditional probability given all other analyses. More specifically, to reach the next state of the Markov chain by the collapsed Gibbs sampling, sample s'_1 given $\mathbf{S}_{-1}$, then go to $\{s'_1, s'_2, s_3, ..., s_N\}$ and so on until $\{s'_1, s'_2, ..., s'_N\} = \mathbf{S}'$. It can be shown that this sampling process defines a Markov chain on $\mathbf{S}, \mathbf{S}', \mathbf{S}'', ...$ whose stationary distribution is the joint posterior distribution $P(\mathbf{S}, \mathbf{W})$, regardless of the initialization of the starting state.

3.5.3 Experiments

Following the experiment framework described in Section 3.2.3, we also evaluate Gibbs sampling with both the type-based and token-based criteria. The learning curves are shown in Figure 3.7 and 3.8, for type-based and token-based evaluation respectively.

As discussed in Section 3.3.3, besides the type-based input, we may also run Gibbs sampling over token-based input. As shown with both the type-based and token-based evaluation, the from of the input data to Gibbs sampling does matter, the same pattern as we have already seen for EM. Also, with the same strategy as for EM, given a list of suffixes, we may allow only the divisions of the word that produces the suffixes in the given

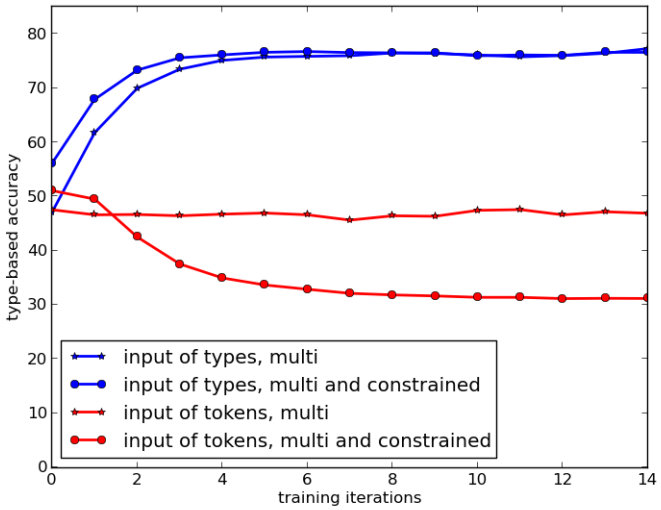


Figure 3.7: The type-based accuracy of Gibbs sampling with multinomial models.

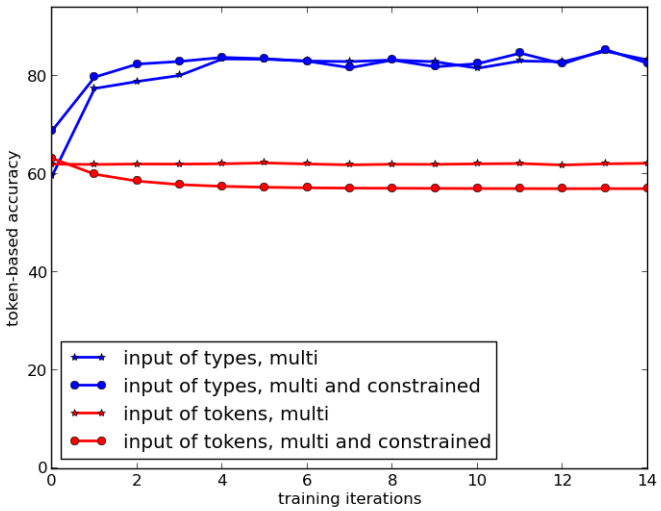


Figure 3.8: The token-based accuracy of Gibbs sampling with multinomial models.

list as legal segmentations. As shown in both Figure 3.7 and 3.8, constraining possible segmentations does speed convergence, but the level of performance is not affected with either type-based or token-based evaluation, again, the same pattern as of EM. It is not a

	type-based acc.	token-based acc.
rule-based with acquired stems (conservative)	69.75%	73.98%
rule-based without acquired stems (greedy)	79.01%	49.55%
EM with multinomial distributions	**82.19**%	**83.50**%
Gibbs sampling with multinomial distributions	79.98%	81.06%

Table 3.6: Morphology learning with type-based input.

surprise at all to see such similar learning patterns of EM and Gibbs sampling, since they only differ in their way to deal with data augmentation.

In Table 3.6, we summarize the models we have discussed so far. Gibbs sampling outperforms the rule-based models but is worse than EM for this experiment. This degree of difference in performance between EM and Gibbs sampling does not decide which algorithm should receive more consideration. The most important thing to read from this table is that none of these models handles token-based input well, so we report only the results of these models running over type-based input. As noticed in (Goldwater et al., 2006), even though not receiving much attention in previous work on morphology learning, token-based input is still worth exploration, since it reflects the real distribution of word frequency. In the next section, we will discuss the long-tail distribution of word frequency, and propose a new model which learns from token-based input in Section 3.7.

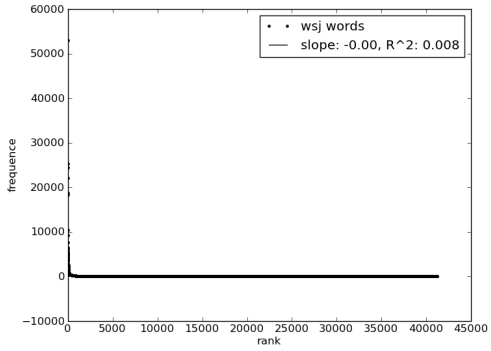
3.6 Long-tail distributions

3.6.1 Plotting long-tail distributions

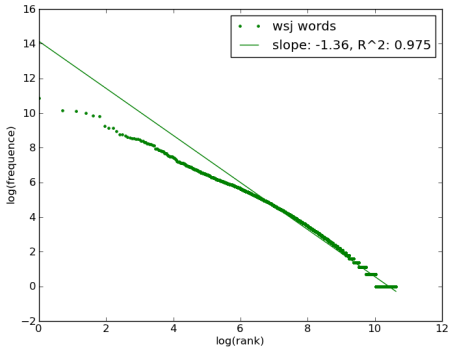
rank	frequency	word type
1	52963	the
2	25285	of
3	24460	to
...		
16749-22699	2	zurn ...
22700-41233	1	zygmunt ...
total word types: 41233		
total word occurrences: 907777		

Table 3.7: Rank words by word frequency in the whole Penn Treebank WSJ corpus. Punctuations are excluded as words. All text strings are lower-cased.

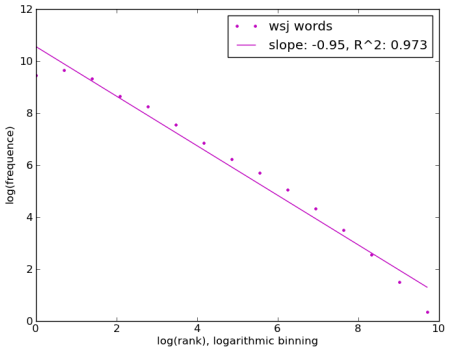
Given a certain corpus, we can compute the word frequency of each word type by counting its occurrences in the corpus. Words in a corpus can then be ranked according to their word frequencies. For example, as computed from Penn Treebank WSJ corpus (Marcus et al., 1993) and shown in Table 3.7, the most frequent word is *the* occurring 52963 times and the second most frequent word is *of* with a count of 25285. If we plot word frequency against word rank, as shown in Figure 3.9-a, there is a long tail of the curve corresponding to the large number of words that occur in low frequency. For example, as shown in Table 3.7, there are 18534 types of words that only occur once in the whole WSJ corpus, and 5950 types occur twice, thus nearly 60% of the total word types occur in low frequency, once or twice. More formally, if a large portion of the population are composed of low-frequency events, forming a longer tail on a rank-frequency plot than normal (Gaussian) distributions, the corresponding distribution is considered as long-tail.



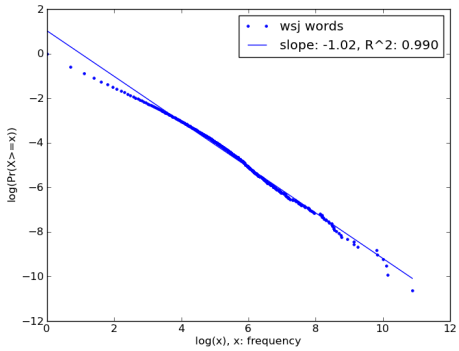
(a) The rank-frequency plot.



(b) The log-log rank-frequency plot.



(c) Logarithmically binning log-log rank-freq.



(d) The cumulative distribution function.

Figure 3.9: Rank-frequency plots of words in WSJ Penn Treebank.

If we plot word frequency against word rank on logarithmic scales, as shown in Figure 3.9-b, it behaves like a straight line, except for the noisy part at the right-hand end of the curve. When we see such a characteristic straight-line on logarithmic scales, it usually suggest that a power law is observed, i.e.

$$f(x) = Cx^{-\alpha}, \qquad \text{constants } C > 0 \text{ and } \alpha > 0.$$

Especially, when we study vocabulary distributions, word frequency is usually said to follow the Zipf's law (Zipf, 1949), which states that (word) frequency is inversely proportional

to its rank, i.e. the exponent α is close to unit. For example, the ratio between frequencies of word *the* and *of*, with rank 1 and 2 respectively, is 2.09, fits Zipf's assumption perfectly. However, the ration between frequencies of word *the* and *to*, with rank 1 and 3 respectively, is 2.16, not very close to 3 as predicted by Zipf's law. Thus we need careful plotting of the data to show its characteristics as we expect.

As shown in Table 3.7, there are many words occur only once or twice. It explains the pattern of the right-hand end in Figure 3.9-b. For low frequency values, there are several wide intervals of ranks in which the points form horizontal lines. So as to let words of the same or close frequency values to be represented by one single point, we need to bin the ranks. For each bin, we normalize the frequencies by the width of the bin they fall in. That is, for the i_{th} bin of interval $[r_i, r_i + 1)$, we compute the corresponding averaged frequency as $\frac{\sum_{r_i \leq r < r_i+1} N_r}{r_{i+1} - r_i}$, where N_r is the count of the r_{th} most frequent word. Moreover, so as to obtain constant widths of bins on logarithmic scales and to increase the width of bins near the end of the curve more than those at beginning of the curve, we create bins with a constant multiplier m such that each bin is m times wider of the one before it. When the multiplier is fixed as 2 and the first bin is [1, 2), the following bins are created as [2, 4), [4, 8), [8, 16) and so forth. As shown in Figure 3.9-c, logarithmically binning does help smooth the data. but a lot of information of individual frequency values is missing.

Instead of rank-frequency plots, we can also draw the Cumulative Distribution Function (CDF). Let F be the discrete random variable that denotes word frequency, and N_r be the count of the r_{th} most frequent word. If there is no tie in ranking, which is true for frequent

words, i.e. true for large N_r, then there are r values of F that are greater or equal to N_r, i.e.

$$Pr[F >= N_r] \sim \frac{1}{Z}r, \qquad Z \text{ a normalizing constant},$$

where $f(x) \sim g(x)$ represents that the limit of the ratio goes to 1 as x grows large. If the frequency distribution follows a power low, i.e. $N_r = Cr^{-\alpha}$, then $Pr[F >= Cr^{-\alpha}] \sim \frac{1}{Z}r$. With the change of variables, we get

$$Pr[F >= f] \sim C_2 f^{-\beta}, \quad \beta = 1/\alpha,$$

where C_2 is a constant. Therefore, if F is of a power-law distribution, its CDF also has a power-law form [7]. Thus, in a log-log plot of CDP, a straight line is also expected, as shown in Figure 3.9-d. Especially, if the frequency distribution follows Zipf's law, i.e. $\alpha = 1$, then $\beta = 1/\alpha = 1$, i.e. the slope of the log-log plot of CDF is also close to unit.

For each log-log plot in Figure 3.9, the data are fit into straight lines by least-square linear regression. Correspondingly, slope α and coefficient of determination R^2 are com-

[7] If we know that CDF follows a power law, it is easier to infer that the corresponding probability distribution function (PDF) also has a power-law form, since PDF is the derivative of CDF. However, given that the PDF is power-law, we cannot simply state that CDF is also power-law, because the integral fails to converge when $\alpha \leq 1$. For example, the following formula in (Newman, 2005) is safe with the assumption of $2 \leq \alpha \leq 3$ in their context,

$$P(x) = C \int_x^\infty x'^{-\alpha} dx' = \frac{C}{\alpha - 1} x^{-(\alpha-1)},$$

but it is not safe in our context since we are especially interested in the case of $\alpha = 1$. This discussion of the power-law PDF and CDF is similar to the discussion in a tutorial by (Lada, 2002). However, with ties in ranking, it is inaccurate to state $Pr[F >= N_r] = \frac{1}{Z}r$. Thus not following either this work, we have given our own induction on the statement that CDF is power-law if probability function is power-law.

puted. For the original log-log rank-frequency plot in Figure 3.9-b, the approximated slope is -1.36, i.e. $\alpha = 1.36$, and $R^2 = 0.975$. After smoothing by logarithmically binning, as shown in Figure 3.9-c, the slope is -0.95 and $R^2 = 0.973$. Finally, in the log-log plot of CDF, the slope is -1.02, thus $\alpha = 1/1.02 = 0.98$, which is the closest to 1. Also the coefficient of determination $R^2(= 0.99)$ is higher with this plot of CDF. Therefore, in the following discussions of other data, we will only use the log-log pot of the corresponding CDF to reveal the power-law form. Moreover, the terms 'power-law' and 'Zipf's law' will be used interchangeably from now on, i.e. the assumption of $\alpha = 1$ is implicitly made.

3.6.2 Log-normal distributions

Even though Zipf's law is massively used to analyze word frequency, long-tail distributions can also be analyzed by log-normal distributions, whose logarithms are normally distributed. When analyzed by log-normal distributions, frequency against rank satisfies

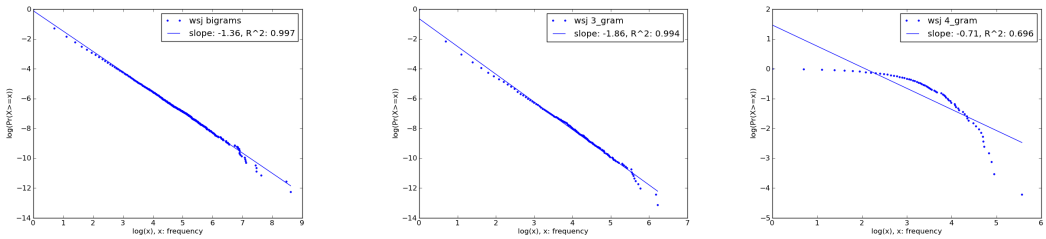
$$f(r) = \frac{1}{r\sigma\sqrt{2\pi}}\, e^{-\frac{(\ln r - \mu)^2}{2\sigma^2}}, \quad r > 0$$

where μ is the mean and σ the standard deviation of the normally distributed $\log(f)$.

If a random variable has a log-normal distribution and especially its variance is large, then in a log-log plot of its CDF, the behavior will also appear to be nearly a straight line for a large portion of the body of the distribution, as shown in Figure 3.10 and 3.11.

Furthermore, we can take advantage of the multiplicative property of log-normal distributions, which states that the product of two log-normal random variables is also log-normally distributed. For example, given that word frequency is log-normal, we can pre-

dict that bigrams are log-normal, trigrams are log-normal and ngrams are log-normal as well. As shown in Figure 3.10, for both bigrams and trigrams on log-log plots, the CDF behaves a straight line, thus can be analyzed by either log-normal or power-law. However, for 4-grams, only part of the curve behaves straightly, so log-normal fits better here.

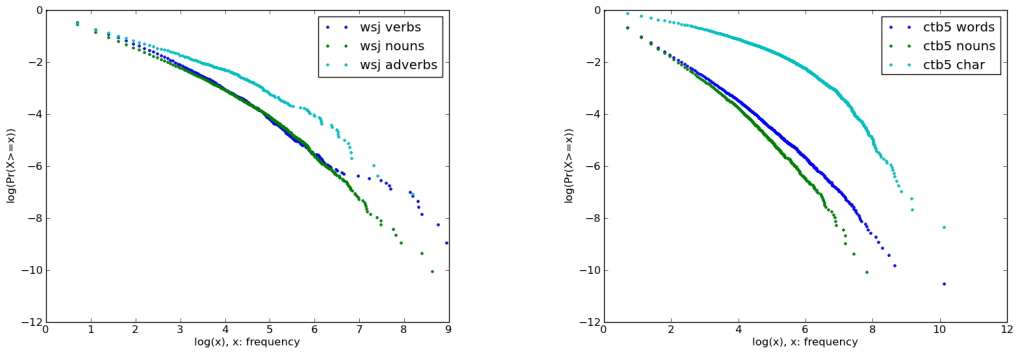


(a) The log-log CDF plot of bigram frequency.

(b) The log-log CDF plot of trigram frequency.

(c) The log-log CDF plot of 4-gram frequency.

Figure 3.10: Distributions of bigrams, 3-grams and 4-grams in the WSJ corpus.



(a) English nouns, verbs and adverbs.

(b) Chinese words, nouns and characters.

Figure 3.11: Observing long-tail distributions for lexical categories in English and Chinese.

Besides word frequency, we also observe long-tail distributions for individual lexical categories, such as verbs, nouns and adverbs, as shown in Figure 3.11-a. Besides English,

we also observe long-tail distributions for Chinese words, nouns and characters, as shown in Figure 3.11-b, which are computed from Chinese Treebank 5.0. Even though, with a glance at the shape of these curves, log-normal distributions seem fits better, it is still hard to say which model works better. Before stating our reason to use log-normal distributions in our morphology learning experiments in Section 3.7, we are going to review some generating processes that generate power-law data or log-normal data in the following section.

3.6.3 Generating power-law and log-normal distributions

Based on the idea of preferential attachment, i.e. a 'richer-get-richer' process, we can generate long-tail distributions. More specifically, if we generate new word occurrences more likely of popular word types in previous process than rarely seen word types, then word frequency of the generated data may exhibit Zipf's law or log-normal distributions. Suppose that we are given i words for a start, $i \geq 1$. Let n_k^i denote the number of occurrences of all the words that occur exactly k times in the previous i words. Let $Pr(w_i = k)$ denote the probability that the $i_{\texttt{th}}$ occurrence is a word that has already appeared k times in the previous $i - 1$ words. Consider the following process as described by Simon (1955),

$$Pr(w_i = k) = \alpha n_0 + F_{i-1} n_k^i,$$

where n_0 and α are constants. If $F_{i-1} = \frac{(1-\alpha)}{i-1}$, then asymptotically $Pr(w_i = k)$ will approach a pow-law distribution. On the other hand, if the constant item is removed from the above process, i.e.

$$Pr(w_i = k) = F_{i-1} n_k^i,$$

where F_i are independent and identically distributed variables with finite mean and variance, then asymptotically $Pr(w_i = k)$ will approach a log-normal distribution.

A even more naive generating process for Zipf's law is Miller (1957)'s monkey, who can not only type with a keyboard, but also distinguish space bar from other keys. If Miller's monkey manages to hit the space bar with a constant probability and never hits the space bar twice subsequently, then the word frequency in the monkey's output follows a power law. One crucial assumption in Miller's demonstration is that all non-space letters are hit with equal probability. In the case that any two letters may be hit with different probabilities, Perline Perline (1996) argues that for all words of length up to a constant, their frequency-rank distribution converges to a log-normal distribution.

After reviewing a brief history of generating processes for power-law and log-normal distributions, Mitzenmacher (2004) suggests that *"It might be reasonable to use which ever distribution makes it easier to obtain results."* As also pointed out in (Mitzenmacher, 2004), if a power-law distribution can have infinite mean and variance, it is inaccurate to analyze it as log-normal. In present examples, we assume $\alpha > 0$ in $f(x) = Cx^{-\alpha}$, thus it is safe for us to assume either distribution. In the following section , for the sake of building proper statistical models, we assume log-normal distributions for morphology learning, because the assumption of power-law distribution does not quite work through.

3.7 Gibbs sampling from log-normal distributions

In both Section 3.4 and 3.5, we assume stems, suffixes and segmentations are multinomially distributed. With this assumption, as we have seen in Section 3.4.2 and 3.5.3, the inference by either EM or Gibbs sampling does not converge over token-based input. We certainly can pre-process token-based input into type-based so that the algorithms can work; however, the frequency information in token-based input will be lost. We have discussed in the above section that, word frequency exhibits a long-tail distribution in real text data, and we agree with Goldwater et al. (2006) that this special distribution of word frequency should be captured in a generative model. After all, for both the EM algorithm and Gibbs sampling, we only assume multinomial distribution so as to take advantage of its simplicity in computation. For EM, the MLE solution has a simple closed-form; and for Gibbs sampling, Dirichlet parameters can be integrated out. If we switch to a distribution modeling long tails, we will need one that also aids the computation.

Word frequency is considered as exhibiting Zipf's law in (Goldwater et al., 2006), following a traditional convention in literature. So as to generate power-law distributions, they propose a Bayesian model composed of two successive generating processes. On the other hand, as discussed in the above section, we argue that there is no theoretical aspect favoring Zipf's law over log-normal distributions in vocabulary study. Thus, to capture the long-tail distribution of word frequency in morphology model, there are at least four alternative models worth consideration: non-Bayesian model for power-law; non-Bayesian model for log-normal; Bayesian model for power-law; and Bayesian model for log-normal. Even

though, for the last experiment in Section 3.5.3, Gibbs sampling performs a little worse than EM, this degree of difference in performance does not decide our choice of inference algorithms in practice. In fact, one main reason for MCMC methods to become so popular, even given the kind of success EM has gained in a long history, is because replacing maximization with simulation for data augmentation reduces the required computation, which may not even be computationally tractable. Besides, Bayesian models for power-law distributions have been studied in (Goldwater et al., 2006), therefore, in this section, we will explore the last alternative, employing a Bayesian inference method, Gibbs sampling, to learn from log-normal data. Furthermore, compared to power-law distributions, we found the following characteristics of log-normal distributions are very useful:

- the conjugate priors of log-normal distributions are known,
- the product of two log-normal random variables is also log-normal,
- log-normal are more flexible to model scatter points.

3.7.1 The probability model

A variable x has a log-normal distribution if $\log(x)$ is normally distributed. The probability density function of a log-normal distribution is:

$$f(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x-\mu)^2}{2\sigma^2}}, \quad x>0,$$

where μ is the mean and σ the standard deviation of the normally distributed logarithm of the variable, and are usually referred as the location and scaler parameter on the logarithmic

scale. Assume that stems t and suffixes f are generated from log-normal distributions with parameters (μ_T, σ_T) and (μ_F, σ_F) respectively, then we we have

$$t|(\mu_T, \sigma_T),\ f|(\mu_F, \sigma_F) \sim \text{Log-normal}(\mu_T, \sigma_T),\ \text{Log-normal}(\mu_F, \sigma_F).$$

So as to have conjugacy, we assume the priors, logarithm of mean $\log(\mu)$ and precision τ (=$1/\sigma^2$), are generated by a Normal-Gamma process with hyperparameters m, p, α, β, i.e.

$$\log(\mu_T),\ \tau_T \sim \text{Normal-Gamma}(m_T, p_T, \alpha_T, \beta_T)$$

$$\log(\mu_F),\ \tau_F \sim \text{Normal-Gamma}(m_F, p_F, \alpha_F, \beta_F)$$

Again, given our independence assumption,

$$P(s = (t, f)|w, \mu_T, \tau_T, \mu_F, \tau_F) = \mathbb{1}(w = t.f)P(t|\mu_T, \tau_T)P(f|\mu_F, \tau_F), \tag{3.7.1}$$

where $\mathbb{1}(w = t.f)$ takes on value 1 if concatenation $t.f$ forms word w, otherwise 0.

3.7.2 Sampling process

By Gibbs sampling, we alternatively sample all variables specified in the probability model, conditioned on the current values of all other variables and the data. In our case, denote $\mathbf{S}_{-i}$ the segmentations of all words but the i_{th} word, i.e. $\mathbf{S}_{-i} = \{s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_N\}$. According to the probability model defined above, for each word w_i, we need to first generate parameters by the Normal-Gamma process conditioned on $\mathbf{S}_{-i}$, and then generate segmentation s_i from log-normal distributions conditioned on the current parameters.

More specifically, given a fixed precision and constant priors, the posterior probability

of mean can be computed as follows,

$$\log(\mu_T) \sim \text{Normal}(\frac{N\bar{N_T} + m_T p_T}{N + p_T}, ((N + p_T) \cdot \tau_T)^{-\frac{1}{2}}),$$
$$\log(\mu_F) \sim \text{Normal}(\frac{N\bar{N_F} + m_F p_F}{N + p_F}, ((N + p_F) \cdot \tau_F)^{-\frac{1}{2}}),$$

where N is the number of samples, $\bar{N_T}$ the sample mean of stem counts, and $\bar{N_F}$ the sample mean of suffix counts in $\mathbf{S}_{-i}$. Then given a fixed mean and constant priors, the posterior probability of precision can be computed as follows,

$$\tau_T \sim \text{Gamma}(\alpha_T + \frac{N}{2}, (\beta_T^{-1} + \frac{1}{2}\sum_{i=1}^{N}(N_{t_i} - \bar{N_T})^2 + \frac{p_T N(\bar{N_T} - \mu_T)^2}{2(p_T + N)})^{-1}),$$
$$\tau_F \sim \text{Gamma}(\alpha_F + \frac{N}{2}, (\beta_F^{-1} + \frac{1}{2}\sum_{i=1}^{N}(N_{f_i} - \bar{N_F})^2 + \frac{p_F N(\bar{N_F} - \mu_F)^2}{2(p_F + N)})^{-1}).$$

Finally, we sample segmentations for each word with the following conditional probability,

$$P(s = (t, f)|w, \mathbf{S}_{-1}, \mu_T, \tau_T, \mu_F, \tau_F) = \mathbb{1}(w = t.f) \times \mathrm{f}_{\mu_T,\tau_T}(r_t) \times \mathrm{f}_{\mu_F,\tau_F}(r_f), \quad (3.7.2)$$

where r_t and r_f denote the frequency rank of stem t and suffix f respectively, and $\mathrm{f}_{\mu,\tau}$ computes the log-normal probability with location and precision parameters μ, τ.

3.7.3 Experiments

Following the experiment framework described in Section 3.2.3, we also evaluate log-normal model with both the type-based and token-based criteria. The learning curves are shown in Figure 3.12 and 3.13, for type-based and token-based evaluation respectively.

We can constrain possible segmentations of a word, by allowing only the divisions that produces the suffixes in a given list as legal segmentations. Even though this trick does

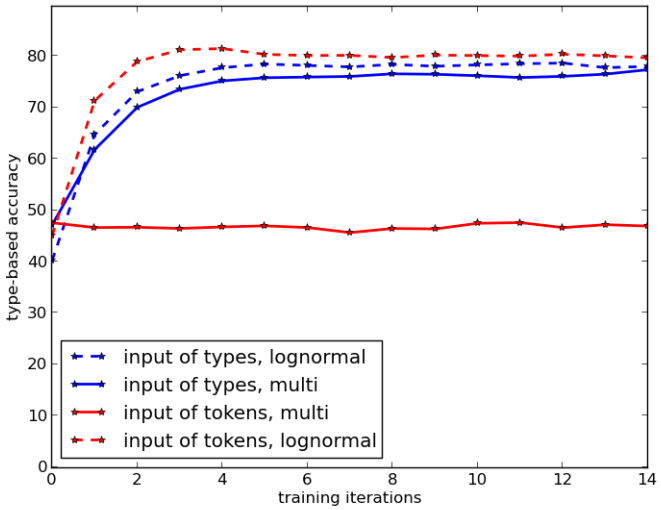


Figure 3.12: The type-based accuracy of Gibbs sampling with log-normal models.

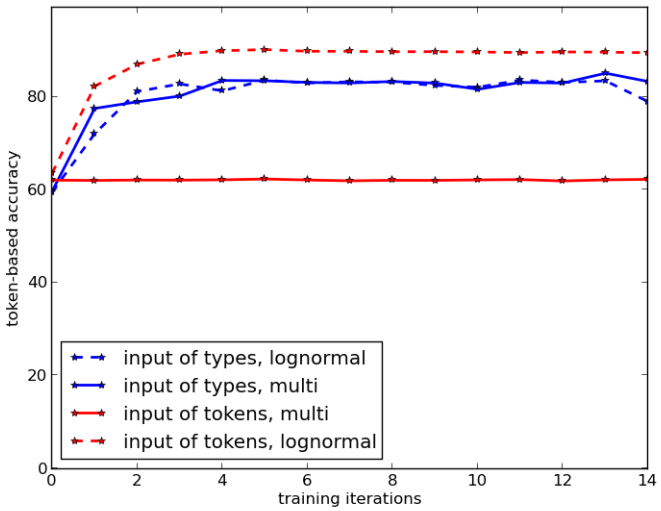


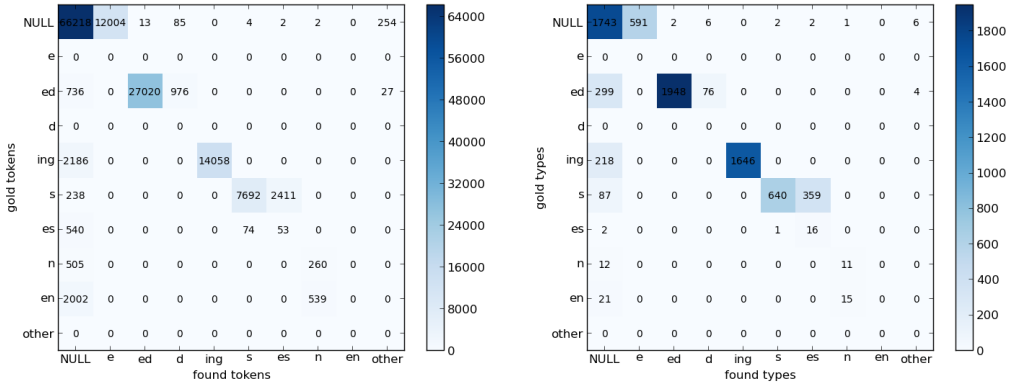
Figure 3.13: The token-based accuracy of Gibbs sampling with log-normal models.

speed convergence, as for the multinomial model, the level of performance is not affected. Therefore, for the sake of clarity, we are not showing this variation in above figures. The most important variation we show in Figure 3.12 and 3.13, is the different forms of input.

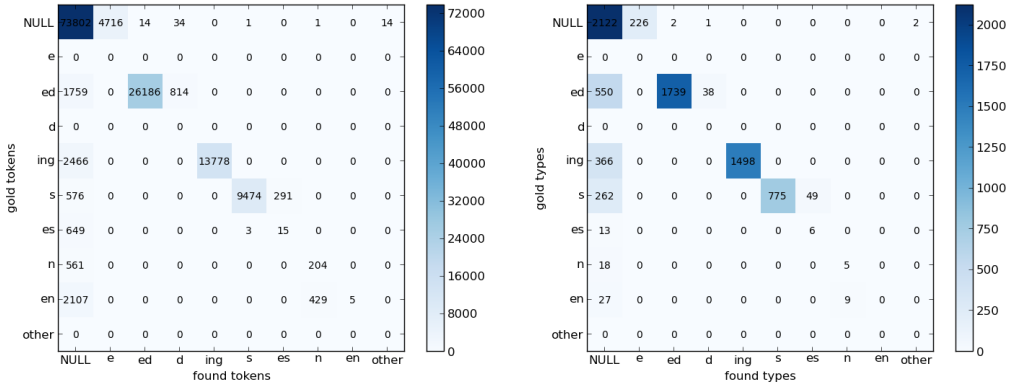
As discussed in Section 3.3.3, besides the type-based input, which contains distinct word types only, we can also run the proposed model over token-based input, which is the text data that reflects the real distribution of word frequency.

Recall that, none of the models that have been discussed so far successfully learns from token-based input, but finally, as shown in Figure 3.12 and 3.13, the proposed log-normal model can take advantage of the form of input that reflect real word frequency distribution. When the input is type-based, both the multinomial model and the log-normal model perform indistinguishably well with either the type-based or token-based evaluation. Whereas, when the input is token-based, the multinomial model fails and the log-normal model performs the best with either the type-based or token-based evaluation. Moreover, when the evaluation is also token-based, the log-normal model takes a great advantage of handling token-based input, distinctively outperforms its own running over type-based input. We compare the errors of log-normal models trained over token-based or type-based input, The confusion matrices for the log-normal model trained over type-based input is shown in 3.14-a and the confusion matrices for token-based input is shown in 3.14-b.

In Table 3.8, we summarize the models we have discussed so far for morphology learning. The rule-based models don't show attractive results in this experiment; however, as shown in (Zhao and Marcus, 2011) and (Zhao and Marcus, 2012b), when learning from the whole WSJ corpus, the rule-based model achieves rather good performance comparable to the state-of-art. It is to our surprise that the EM algorithm is not applied to morphology learning as widely as we expect. But this (generally) widely used algorithm shows its strength again in our experiments. When the evaluation is type-based, EM performs



(a) Log-normal over type-based input.



(b) Log-normal over token-based input.

Figure 3.14: Confusion matrices for log-normal models over different forms of input.

the best. The problem is, when multinomial distributions of stems, suffixes and segmentations are assumed for the sake simplicity in computation, neither the EM algorithm nor the Gibbs sampling learns from token-based input. As noticed in (Goldwater et al., 2006), token-based input usually does not bother researchers on morphology learning, but we agree with them that the real distribution of word frequency should be captured. Goldwater et al. (2006) introduced an additional generating process to transform the multinomial sam-

	type-based acc.		token-based acc.	
	types input	tokens input	types input	tokens input
rule-based (conservative)	69.75%	-	73.98%	-
rule-based (greedy)	79.01%	-	49.55%	-
EM for multinomial	**82.19**%	-	83.50%	-
Gibbs for multinomial	79.98%	-	81.06%	-
Gibbs for log-normal	79.72%	77.89%	83.61%	**89.53**%

Table 3.8: Morphology learning with either type-based or token-based input.

ples to exhibit Zipf's law; thus even though the token-based input can be generated by their model, the overall performance is not improved. Instead, we first argues that it is not necessary to follow the convention of analyzing word frequency with Zipf's law (or power-law). Then we propose a log-normal model that directly generates long-tail distributions. So as to avoid mathematically complex computations, we use Gibbs sampling for inference. Finally, our effort on capturing the real distribution of word frequency pays back and leads to the best model with token-based evaluation.

3.8 related work

Our main focus of this work is to capture a linguistic aspect for improving morphology learning. This is a fundamental effort in work on morphology learning, and actually, compared to supervised work, unsupervised work pay much more attention to capturing

linguistic aspects in their models. The most popular way to incorporate more linguistic features is to add more components to the probability model or generative process. For example, in a widely used benchmark Morfessor (Creutz and Lagus, 2007), the probability of a morpheme lexicon is the product of three factors: 1) the prior probability that the lexicon is of the exact size; 2) the joint probability of generating all the morphemes in the lexicon; and 3) the number of permutations of this lexicon. Furthermore, the generating probability of each morpheme, is the product of form probability and usage probability, each of which is of multiple components that can be decomposed further. In this way, properties of each morpheme, such as frequency, length, left or right perplexity and so on are carefully considered. Similarly, when syntactic context is considered to help morphology learning in (Lee et al., 2011), besides the basic lexicon and segmentation model, the generative process also contains a model capturing the dependencies between the syntactic categories of adjacent words and a model capturing morphological agreement between adjacent segmentations.

So as to leverage arbitrarily overlapping features, Poon et al. (2009) proposed a log-linear model for morphology learning and achieves the state-of-the art performance. It is also not uncommon to introduce a special step for massaging the intermediate output, for example, Lignos et al. (2009) proposed a post-processing step to break compound words after the induction of morpheme lexicons. And even though we only tried two simple strategies, the decision process to segment words with acquired morphemes could be rather complex as in (Dasgupta and Ng, 2007), carefully designed with linguistic knowledge on morphology. Compared to these efforts, we are not only trying to capture the signature distributions of natural language, but also challenging the conventional linguistic story about

it, making the modeling of this linguistic aspect more straightforward and improving performance.

The experimental framework of this chapter closely follows Goldwater et al. (2006)'s experiments on capturing the long-tail distribution of word frequency in morphology learning. Word frequency is considered as exhibiting Zipf's law in (Goldwater et al., 2006), following a traditional convention in literature. So as to generate power-law distributions, they propose a Bayesian model composed of two successive generating processes. A generalized Chinese restaurant process (Aldous, 1985), Pitman-Yor process (Ishwaran and James, 2003) is exploited for producing power-law distributions in their work. Pitman-Yor process is also based on the principle of preferential attachment (discussed in Section 3.6.3), moreover, the outcomes remain exchangeable (Pitman and Yor, 1997), i.e. the ordering of events does not affect their cumulative probability, and only the number of events of each type does. This generating process is used as an 'adaptor' for transforming outcomes of any morphology model to exhibit Zipf's law. The morphology model they experiment with is exactly the one we described in Section 3.5, Gibbs sampling for multinomial.

We have experimented with both Bayesian and non-Bayesian approaches for morphology learning in this work, even though Bayesian approaches become more and more popular recently. Even in earlier works that are not labeled as Bayesian, the Bayesian notion of probability is quite often assumed, since one's assumption of morphology model usually affects the learning. For example, in a widely-used benchmark, Linguistica (Goldsmith, 2001), the principles of the minimum description length (MDL) framework is invoked, so that an morphology model is not only judged by its ability to approximate data, but also on

its complexity as a theory. This MDL framework is equivalent to a maximum a posteriori (MAP) model, which is used to estimate the language model in another widely-used benchmark, Morfessor (Creutz and Lagus, 2007). However, MAP estimation is not considered as a representative Bayesian method, since MAP estimates are point estimates. In more recent works, such as (Goldwater et al., 2006; Lee et al., 2011; Chahuneau et al., 2013) etc., Markov Chain Monte Carlo methods, such as Gibbs sampling, are used to simulate posterior distributions. By the way, we figure out the full posterior analysis of Normal distributions with the help of a lecture note by Jordan (2010).

Finally, we are not the only one trying to capture the long-tail distribution of word frequency in morphology learning, even though we seem the only one managing to take advantage of this distribution. Besides the aforementioned work by Goldwater et al. (2006), Chahuneau et al. (2013) also uses Pitman-Yor processes for their language model as power-law generators. Creutz (2003) incorporates the Zipf's law in their morphology model once, but not in their following works. Moreover, as discussed in Chapter 2, the bootstrapping algorithm proposed to acquire functional elements, including morphological endings for English, is motivated by Chan (2008)'s work on morphology learning. Zipf's law in morphology is considered as the basic observation that motivates Chan (2008)'s work, however, we didn't introduce our algorithm based on the understanding of any particular distribution.

3.9 Conclusion

Morphology learning is a widely studied topic and in this work, we focus on capturing long-tail distributions in morphology models, and even manage to take advantage of this signature distribution of natural language, improving the performance by a significant margin for a token-based evaluation. Compared to previous attentions to some linguistic aspect in morphology learning, we are not only trying to capture it but also challenging the conventional story about it, so as to really take advantage of the linguistic aspect of interest.

So as to deal with real text input, which reflects long-tail word distribution, we have been open to rule-based methods, maximum likelihood methods and Bayesian methods. And only the last proposed Bayesian model with log-normal assumptions handles token-based input well. Even though previously proposed Bayesian models that generate power-law distributions can also explain off word frequencies, our proposed model is the first one that actually takes advantage of word frequencies for a toke-based evaluation and performs better with real text input than with type-based input. Since word distribution is convectionally studied by power-law distributions, we have devoted a section to examine whether there is any theoretical aspect favoring Zipf's law over log-normal distributions in vocabulary study, and have discovered none. Thus, our contribution is not to report on a successful play with Bayesian inference for another mathematically complex model, but is our willingness to dive deep enough in linguistic theories and adapt them for our engineering use.

Chapter 4

Determinism in POS tagging and Chinese word segmentation

4.1 Introduction

In [8] recent work, interesting results are reported for applications of integer linear programming (ILP) such as semantic role labeling (SRL) (Roth and Yih, 2005), dependency parsing (Martins et al., 2009) and so on. In an ILP formulation, 'non-local' deterministic constraints on output structures can be naturally incorporated, such as "*a verb cannot take two subject arguments*" for SRL, the projectivity constraint for dependency parsing and so on. In the contrast of probabilistic constraints that are estimated from training examples, this type of non-local constraints is usually hand-written reflecting one's linguistic knowledge.

Dynamic programming techniques based on Markov assumptions, such as Viterbi de-

[8] This chapter extends (Zhao and Marcus, 2012a) with elaborations but little extra work.

coding, cannot handle those 'non-local' constraints as discussed above. However, it is possible to constrain Viterbi decoding by 'local' deterministic constraints, e.g. "*assign label t to word w*" for POS tagging. This type of constraint may come from human input solicited in interactive inference procedure (Kristjansson et al., 2004).

In this work, we explore deterministic constraints for two fundamental NLP problems, English POS tagging and Chinese word segmentation. We show by experiments that, with proper representation, large number of deterministic constraints can be learned automatically from training data, which can then be used to constrain probabilistic inference.

For POS tagging, the learned constraints are directly used to constrain Viterbi decoding. The corresponding constrained tagger is 10 times faster than searching in a raw space pruned with beam-width 5. Tagging accuracy is moderately improved as well. For Chinese word segmentation (CWS), which can be formulated as character tagging, analogous constraints can be learned with the same templates as English POS tagging. High-quality constraints can be learned with respect to a special tagset, however, with this tagset, the best segmentation accuracy is hard to achieve. Therefore, these character-based constraints are not directly used for determining predictions as in English POS tagging. We propose an ILP formulation of the CWS problem. By adopting this ILP formulation, segmentation F-measure is increased from 0.968 to 0.974, as compared to Viterbi decoding with the same feature set. Moreover, the learned constraints can be applied to reduce the number of possible words over a character sequence, i.e. to reduce the number of variables to set. This reduction of problem size immediately speeds up an ILP solver by a factor of 100.

In the next section, we are going to explore deterministic constraints for English POS

tagging. In Section 4.3, we explore deterministic constraints for Chinese word segmentation. We review related work in Section 4.4 and conclude in Section 4.5.

4.2 English POS tagging

It may help to get a preliminary sense of the determinism in POS tagging, if we consider a feature-based representation of POS tags. Suppose that, following (Chomsky, 1970), we distinguish major lexical categories (Noun, Verb, Adjective and Adverb) by two binary features: $+|-$ N and $+|-$ V. Let ($+$N, $-$V) =Noun, ($-$N, $+$V) =Verb, ($+$N, $+$V) =Adjective, and ($-$N, $-$V) =Adverb. As depicted in Table 4.1, this is a much more reduced feature-based analysis of lexical categories, compared to what we proposed in Section 2.7.1. In this simple example illustrating determinism in language, we only show how syntactic features are deterministically imposed by local context. For example, given this feature-based analysis, a word occurring in between a preceding word *the* and a following word *of* always bears the feature $+$N.

Nouns	+N	-V
Verbs	-N	+V
Adjectives	+N	+V
Adverbs	-N	-V

Table 4.1: A (reduced) feature-based analysis of the main lexical categories.

On the other hand, consider the annotation guideline of English Treebank (Marcus et al.,

1993), according to which the POS tag VBG tags verbal gerunds, NNS tags nominal plurals, DT tags determiners and so on. Following this POS representation, there are as many as 10 possible POS tags that may occur in between *the–of*, as estimated from the WSJ corpus of Penn Treebank, including NN, NNS, JJ, VBG etc. At the first glance, the principle of determinism does not necessarily lead to determinacy in POS tagging. However, for a specific word, the context *the–of* is usually enough to decide a tag for it. Therefore, in an engineering-oriented representation, an abstract property such as determinism may not realize itself as generally as in linguistic studies, but we can always try lexicalized models as NLPers usually do.

4.2.1 Templates of deterministic constraints

To explore determinacy in the distribution of POS tags in Penn Treebank, we need to consider that a POS tag marks the basic syntactic category of a word, such as nominal, verbal etc., as well as its morphological features, such as a gerund form, a plural form etc. Thus, a constraint that may determine a word occurrence's POS category should reflect both the contextual and the morphological features of the corresponding word. For example, we observe that a word ending with *-es* and occurring in the context of *the–of* is always tagged with NNS in the WSJ corpus of Penn Treebank.

The practical difficulty in representing such deterministic constraints is that we do not have a perfect mechanism to analyze morphological features of a word. Endings or prefixes of English words do not deterministically mark their morphological transformations. For

example, ending *-s* marks the plural form in word *trades* but not in word *miss*. We propose to compute the morph feature of a word as the set of all of its possible tags, i.e. all tag types that are assigned to the word in training data. For example, the morphological feature of *trades* is computed as {NNS, VBZ}, and the morphological feature of *miss* is computed as {NN, VB, VBP}, avoiding the complexity in morphological analyses. Furthermore, we approximate unknown words in testing data by rare words in training data. For a word that occurs less than 5 times in the training corpus, we compute its morph feature as its last two characters, which is also conjoined with binary features indicating whether the rare word contains digits, hyphens or upper-case characters respectively. For both frequent words and rare words, we show examples of their morphological features in Table 4.2.

(frequent)	(set of possible tags of the word)
w_0=*trades*	m_0={NNS, VBZ}
(rare)	(the last two characters...)
w_0=*time-shares*	m_0={-es, HYPHEN}

Table 4.2: Morphological features of frequent words and rare words.

Furthermore, we consider *bigram* and *trigram* templates for generating potentially deterministic constraints, as described in Table 4.3. Let w_i denote the i_{th} word ($i = -1, 1$) relative to the current word w_0; and m_i denote the morphological feature of w_i. A *bigram* constraint includes the current word w_0 or its morphological feature m_0 as well as one contextual word (w_{-1}or w_1) or its morphological feature (m_{-1} or m_1). A *trigram* constraint includes both contextual words or their morphological features.

bigram	$w_{-1}w_0$, w_0w_1, $m_{-1}w_0$, w_0m_1 $w_{-1}m_0$, m_0w_1, $m_{-1}m_0$, m_0m_1
trigram	$w_{-1}w_0w_1$, $m_{-1}w_0w_1$, $w_{-1}m_0w_1$, $m_{-1}m_0w_1$ $w_{-1}w_0m_1$, $m_{-1}w_0m_1$, $w_{-1}m_0m_1$, $m_{-1}m_0m_1$

Table 4.3: The templates for deterministic constraints of POS tagging.

4.2.2 Learning and decoding of deterministic constraints

Given the templates proposed above, we can learn deterministic constraints by counting and thresholding. In a given corpus, if a constraint c relative to w_0 'always' assigns a certain POS category t^* to w_0, with respect a threshold value thr, i.e.

$$\frac{\texttt{count}(c \wedge t_0 = t^*)}{\texttt{count}(c)} > \texttt{thr},$$

we consider c a deterministic constraint. A cutoff number is also introduced to filter out rarely seen patterns. For example, by the above definition, the constraint ($w_{-1} =$ *the*, $m_0 = \{\texttt{NNS, VBZ}\}$, and $w_1 =$ *of*) is deterministic, and it determines the tag of w_0 to be NNS.

Given the deterministic constraint ($w_{-1} =$ *the*, $m_0 = \{\texttt{NNS, VBZ}\}$, and $w_1 =$ *of*), when we see the word *trades*, whose morph feature is $\{\texttt{NNS}, \texttt{VBZ}\}$, occurring between *the-of*, this occurrence of *trades* should be tagged with NNS. However, there may be more than one constraint invoked by the same sequence of words, and so as to achieve a higher precision, rather than a higher recall, we make a tag decision by deterministic constraints only if all relative constraints agree on the same tag. This way of decoding is purely rule-based and involves no probabilistic inference. And this tagging process only produces tags

where deterministic constraints apply, but doesn't affect those ambiguous contexts.

4.2.3 Search in a constrained space

Following most previous work on supervised POS tagging, we consider tagging as a sequence classification problem and decompose sequence score over the linear structure, i.e.

$$\hat{\mathbf{t}} = \underset{\mathbf{t} \in \mathbf{tagGEN}(\mathbf{w})}{\arg\max} \sum_{i=1}^{n} score(t_i) \tag{4.2.1}$$

where function **tagGEN** maps input sentence $\mathbf{w} = w_1 ... w_n$ to the set of tag sequences that are of length n. A given score function $score(t_i)$ computes the score of tag t_i in a sequence regarding some model. If the number of possible tags for each word is a constant T, the space of **tagGEN** is as large as T^n. On the other hand, if we constrain **tagGEN** by deterministic constraints, i.e. for some words, the number of possible tags is reduced to 1, the search space is reduced to T^m, where m is the number of (unconstrained) words that are not subject to any deterministic constraints.

In practice, brute-force searching the space of **tagGEN** is intractable. Thus dynamic programming techniques are widely used for tagging, e.g. the Viterbi algorithm that runs in $O(nT^2)$. When searching in a constrained space by Viterbi, suppose that among the m unconstrained words, m_1 of them follow a word that has been tagged by deterministic constraints and m_2 (=m-m_1) of them follow another unconstrained word. Viterbi decoder runs in $O(m_1 T + m_2 T^2)$. We compare this constrained search with beam search, a popular pruning technique widely-used for tagging. If we only memorize the top B paths for each state, i.e. beam width is B, then Viterbi runs in $O(nBT)$. As depicted in Figure 4.1, the

constrained search and beam search can perfectly work together, since they are guided by different standards. During a Viterbi beam search with beam B, when there is no deterministic constraint applied, the search algorithm explores each word by expanding only B most promising paths, however, when there is a deterministic constraint applied, no sorting is required for pruning and all paths in the enumeration are simply appended by the same successor state. Moreover, as we are going to discuss in the following section, the deterministic constraints not only constrain the search space, but also provide additional lookahead features to tagging models.

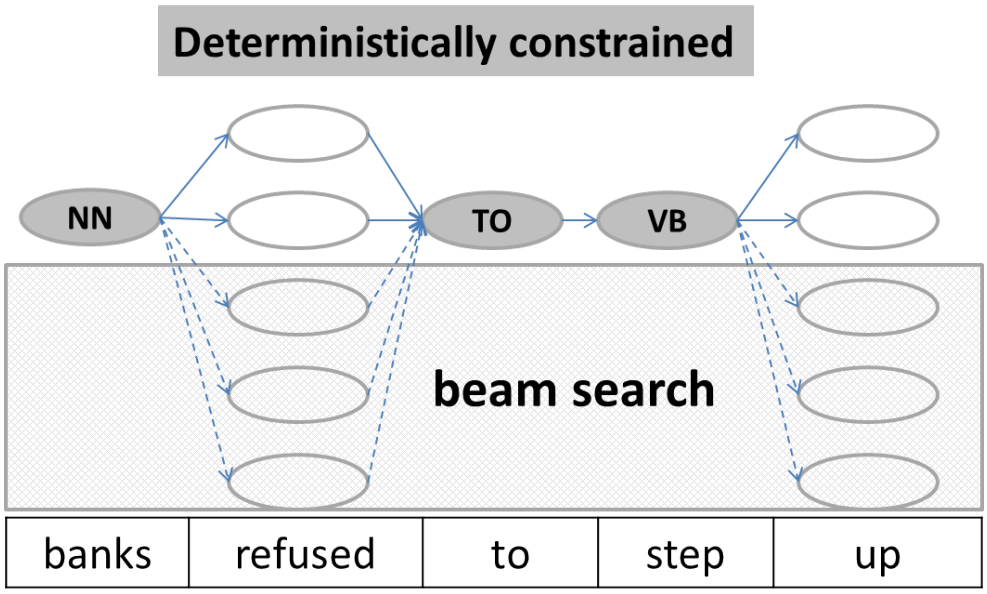


Figure 4.1: Beam search that is also constrained by deterministic constraints.

Lookahead features

The scores of tag predictions are usually computed in a high-dimensional feature space. Besides lexicalized features, the tagging history also constitutes an important part of the

feature space. By the Viterbi algorithm, we may look up the preceding tags in the dynamic path, however, we cannot rely on the categorical information of the following words to predict for the current word. On the other hand, when a Viterbi decoder searches in a constrained space, the following words may have already been tagged by the deterministic constraints, thus our feature space can be extended by lookahead features as well. More specifically, we use the following templates to generate lookahead features:

$$t_0\&t_1, \quad t_0\&t_1\&t_2, \quad \text{and } t_{-1}\&t_0\&t1$$

where t_i denotes the tag of the i_{th} word relative to the current word w_0. Putting these templates together with those used in (Ratnaparkhi, 1996), we replicate the feature set B in (Shen et al., 2007). As discussed in (Shen et al., 2007), categorical information of neighbouring words on both sides of w_0 helps resolve POS ambiguity of w_0. So as to have lookahead features available, Shen et al. (2007) proposed a bidirectional search instead of left-to-right as in Viterbi decoding. In this work, we stick to the more straightforward left-to-right search strategy, but lookahead features are still made available where deterministic constraints are invoked. As we will see in the following section, using both *bigram* and *trigram* templates discussed in Section 4.2.1, the deterministic constraints learnt from the WSJ corpus cover more than 80% of the input tokens. Given that more than 80% words have been tagged when we conduct the Viterbi decoding, the lookahead features made available at this stage do help improve the tagging performance.

	precision	recall	F_1
bigram	0.993	0.841	0.911
trigram	**0.996**	0.608	0.755
bi+trigram	0.992	**0.857**	**0.920**

Table 4.4: POS tagging with deterministic constraints.

4.2.4 Experiments on deterministic predictions of POS tags

In this section, we show how well the learnt deterministic constraints predict POS tags and in the next section, we use these constraints to constrain probabilistic POS tagging. We follow the conventional split of the Penn WSJ corpus, as in (Collins, 2002), (Shen et al., 2007) etc, dividing this corpus into training set (sections 0-18), development set (sections 19-21) and the final test set (sections 22-24). The development set is used to choose training iterations and other parameters, and the experiments in both this section and the next section are done on the final test set.

For English POS tagging, we evaluate the deterministic constraints generated by the templates described in Section 4.2.1. Since this tagging process only produces tags where deterministic constraints apply, leaving alone those ambiguous contexts, we report F-measure for tagging performance. Following the convention, precision p is defined as the percentage of correct predictions out of all predictions, and recall r is defined as the percentage of correctly tagged words out of all words. Then F-measure F_1 is computed by $2pr/(p+r)$.

As shown in Table 4.4, deterministic constraints learned with both *bigram* and *trigram*

m_0=$\{$VBN, VBZ$\}$ & m_1=$\{$JJ, VBD, VBN$\}$ $\rightarrow$ VBN
w_0=also & m_1=$\{$VBD, VBN$\}$ $\rightarrow$ RB
m_0=$-$es & m_{-1}=$\{$IN, RB, RP$\}$ $\rightarrow$ NNS
w_0=last & w_{-1}= the $\rightarrow$ JJ
m_0 = $\{$NNS, VBZ$\}$ & w_{-1} =the & w_1=of $\rightarrow$ NNS
m_0 = $\{$NN, VBP$\}$ & w_{-1} =the & w_1=of $\rightarrow$ NN

Table 4.5: Examples of deterministic constraints for POS tagging.

templates are all very accurate in predicting POS tags for target words. Constraints generated by the *bigram* template alone can already cover 84.1% of the input words with a high precision of 0.993. By adding the constraints generated by *trigram* template, recall is increased to 0.857 with little loss in precision. Since these deterministic constraints are applied before the decoding of probabilistic models, reliably high precision of their predictions is crucial. For all above experiments, we set the cutoff number as 5 and use a threshold value of 0.99 to ensure highly precise predictions.

There are 114589 *bigram* deterministic constraints and 130647 *trigram* constraints learned from the training data. We show a couple of examples of learnt constraints in Table 4.5. A constraint is composed of the preceding word w_{-1}, the current word w_0 and the following w_1, as well as the morph features m_{-1}, m_0 and m_1. For example, the first constraint in Table 4.5 predicts tag VBN for w_0, if the set of possible tags of w_0 contains $\{$VBN, VBZ$\}$ and its following word w_1 has possible tags of $\{$JJ, VBD, VBN$\}$. And as we expect at the beginning of this section, the context *the-of* predicts different nominal tags

for words with different morphological features. Thus we have proposed to calculate the morphological feature of a word to be the set of its possible tags. In such a way, the abstract understanding of determinism in the relationship between the syntactic feature of a word and its local context is instantiated as determinacy in POS tagging.

4.2.5 Experiments on constrained English POS tagging

We replicate the English POS tagger described in (Collins, 2002) as the baseline model in this work, which uses a perceptron like learning schema and the classic Viterbi algorithm for decoding. As discussed in Section 4.2.3, we adopt a very compact feature set for English POS tagging, the one used in (Ratnaparkhi, 1996)[9]. While searching in a constrained space, we can also extend this feature set with some basic lookahead features and replicates the feature set B used in (Shen et al., 2007). For both feature sets, we show the corresponding tagging accuracy by the constrained search using the Viterbi algorithm, which takes the tagging output of the deterministic process as input. With the feature set of (Ratnaparkhi, 1996), we also report for the unconstrained search that takes raw data as input, and use it as a baseline model. However, with the feature set B in Shen et al. (2007) that contains lookahead features, we can only run the constrained search, since without the input of deterministically predicted POS tags, a left-to-right Viterbi decoder cannot look ahead at the following words. Moreover, we vary the beam width, 1 or 5, for beam search. These tagging results are shown in Table 4.6.

[9]The only modification is that we use the lowercase of the current word w_0 instead of w_0. Our implementation of this feature set is basically the same as the version used in (Collins, 2002).

Ratnaparkhi (1996)'s feature		
	Beam=1	Beam=5
raw	$96.46\%/3\times$	$97.05/1\times$
constrained	$96.80\%/14\times$	$97.16/10\times$
Feature B in (Shen et al., 2007)		
(Shen et al., 2007)	97.15% (Beam=3)	
constrained	$97.03\%/11\times$	**97.20**$/8\times$

Table 4.6: POS tagging accuracy and speed. The baseline for speed in all cases is the unconstrained tagger using (Ratnaparkhi, 1996)'s feature and conducting a beam (=5) search.

Improved performance. English (supervised) POS tagging is a very well studied problem, and a English POS tagger with an accuracy above 97% can be easily built in a couple of hours, e.g. the baseline model (beam = 5) in our work. When there is no further input of unlabeled data, the state-of-art tagging accuracy is 97.33% (Shen et al., 2007), which is achieved by using a much more complex feature set than the one replicated here. In this work, we focus on two basic feature sets only, and show that the proposed system achieves a higher performance compared to other systems with the same feature set. As shown in Table 4.6, even without lookahead features, e.g. Ratnaparkhi (1996)'s feature, overall tagging accuracy can be improved by conducting a deterministic process first, during which more than 80% input are confidently tagged with very high precision. When a feature set with lookahead features is used, e.g. Feature B in (Shen et al., 2007), the proposed search algo-

rithm, which adopts the more straightforward left-to-right search strategy, performs better than Shen et al. (2007)'s bidirectional search, achieving a tagging accuracy of 97.20%.

Improved efficiency. The proposed constrained search has even bigger advantage in efficiency than in performance. For example, our baseline model, which conducts a beam (=5) search in a unconstrained space, can be speeded up by reducing the beam width, by searching in a constrained space, or by both. As shown in in Table 4.6, by reducing the beam width from 5 to 1, the system is 3 times fast; by searching in a constrained space, the system is 10 times fast; and by both, the system is 14 times fast. Even with a more complex feature set, searching in a constrained space is still 7 times faster than the baseline model.

NO trade-off between performance and efficiency. Pruning is usually employed when efficiency is the priority to consider but the sacrifice in performance is also acceptable. However, the proposed constrained search has shown to improve the efficiency without hurting the performance. Consider Ratnaparkhi (1996)'s feature set. As shown in Table 4.6, when the beam-width is reduced from 5 to 1, the tagger (beam=1) is 3 times fast but the tagging accuracy is badly hurt. In contrast, the constrained search with beam width 5 is 10 times fast, but the tagging accuracy is even moderately improved, increased to 97.16%.

4.3 Chinese Word Segmentation (CWS)

Given a sequence of Chinese characters with no spaces as delimiters of words, the task of Chinese Word Segmentation (CWS) is to segment this sequence of characters into legal

words. Considering the ambiguity problem that a Chinese character may appear in any relative position in a word and the out-of-vocabulary (OOV) problem that it is impossible to observe all words in training data, research on CWS are active.

In Section 4.3.1, we will first consider a character-based model for word segmentation, in which way CWS can be solved as a sequence labeling problem. In Section 4.3.2, we use the same templates for English POS tagging to generate deterministic constraints for the character-based CWS model, and show that whether highly precise constraints can be learnt is very sensitive to the representation of character tags.

In Section 4.3.3, we consider the word-based model for CWS and then in Section 4.3.4, we propose an Integer Linear Programming (ILP) formulation of the word segmentation problem. As shown with experiments in Section 4.3.5, when the joint inference of CWS with POS tagging is not considered, solving CWS as an ILP problem achieves the state-of-art performance. Furthermore, the deterministic constraints learnt with the character-based CWS model can be applied to constrain word-based CWS models, e.g. the ILP formulation in our work, and improve the efficiency by more than 100 times.

4.3.1 Word segmentation as character-based tagging

The Chinese word segmentation (CWS) problem is widely formulated as a character-based tagging problem (Xue, 2003). The tag of each character represents its relative position in a word. There are two popular tagsets considered in literature for tagging characters,

- `IB`: tag `B` tags the beginning of a word and `I` all other positions;

- `BMES`: tag `B`, `M` and `E` represent the beginning, middle and end of a multi-character word respectively, and `S` tags a single-character word.

For example, with tagset `BMES`, a word can be unambiguously composed by four consecutive characters that are associated with the tag sequence `BMME`. However, with tagset `IB`, four consecutive characters that are associated with a tag sequence `BIII` may compose a word if the following tag is `B`, or only form part of a word if the following tag is `I`. Even though the character-based tagging accuracy is usually higher with tagset `IB`, tagset `BMES` is more popular in use since the corresponding performance of CWS is usually higher, which is, after all, the concerned problem.

A character-based CWS decoder is to find the highest scored tag sequence $\hat{\mathbf{t}}$, given the input character sequence $\mathbf{c}$, the same formulation for POS tagging, i.e.

$$\hat{\mathbf{t}} = \underset{\mathbf{t} \in \mathbf{tagGEN}(\mathbf{c})}{\arg\max} \sum_{i=1}^{n} score(t_i). \tag{4.3.1}$$

4.3.2 Experiments on character-based deterministic constraints

We can use the same templates as described in Table 4.3 to generate potentially deterministic constraints for character-based tagging, except that there are no morphological features computed for Chinese characters. In this section, we are going to evaluate how well these deterministic constraints predict relative positions of Chinese characters. We run experiments of the Chinese word segmentation problem on the Penn Chinese Treebank 5.0. Following (Jiang et al., 2008a), we divide this corpus into training set (chapters 1-260), development set (chapters 271-300) and the final test set (chapters 301-325).

Recall that we discussed two tagsets IB and BMES in the above section. With respect to either tagset, we use both *bigram* and *trigram* templates to learn character-based deterministic constraints, with the same cutoff number 5 and threshold value 0.99. These constraints are then deterministically decoded to predict tags for sequences of Chinese characters. In the experiments of this section, tagging output are evaluated by the F-measure of the tagged character sequences, i.e. the same evaluation with the experiments on English POS tagging, instead of by the F-measure of the word segmentation problem.

As shown in Table 4.7, when tagset IB is used for character-based tagging, highly precise deterministic constraints can be learned. However, when tagset BMES is used, the learned constraints do not always make reliable predictions, and the overall precision is not high enough to constrain a following probabilistic model. Therefore, we will only use the deterministic constraints that predict IB tags in the following CWS experiments. It is interesting but not surprising to notice, again, that the determinacy of a problem is sensitive to its representation. Since it is hard to achieve the best segmentations with tagset IB, we propose an indirect way to use these constraints in the following section, instead of applying these constraints as straightforwardly as in English POS tagging.

tagset	precision	recall	F_1
BMES	0.989	0.566	0.720
IB	**0.996**	**0.686**	**0.812**

Table 4.7: Character-based tagging with deterministic constraints.

4.3.3 Word-based segmentation model

In this section, we are going to describe a word-based segmentation model. The output of a character-based tagging process is a sequence of characters with tags indicating their relative positions in words. And a post-processing step is required for the composition of words according to these output tags. Thus the choice of character-based tagset is sensitive. However, with a word-based model, the output are directly word segmentations, thus no post-processing step is required.

Suppose that function **segGEN** computes all possible segmentations of an input sequence. Given a sequence of characters $\mathbf{c}$, function **segGEN** maps $\mathbf{c}$ to sequences of words that form a segmentation of $\mathbf{c}$. For example, $\mathbf{w} = (c_1..c_{l_1})...(c_{n-l_k+1}...c_n)$ represents a segmentation of k words, and the lengths of the first and last word are l_1 and l_k respectively. A word-based CWS decoder finds the highest scored word sequence $\hat{\mathbf{w}}$ in $\textbf{segGEN}(\mathbf{c})$, i.e.

$$\hat{\mathbf{w}} = \underset{\mathbf{w} \in \textbf{segGEN}(\mathbf{c})}{\arg\max} \sum_{i=1}^{|\mathbf{w}|} score(w_i), \tag{4.3.2}$$

In the contrast of character-based models, a word-based CWS model directly computes output of words, thus no further errors may be made in the transformation of representations. In early work, rule-based models are proposed to find words one by one based on heuristics such as forward maximum match (Sproat et al., 1996). In more recent work, Viterbi-like exact search or beam search are more favorable, e.g. (Zhang and Clark, 2007) and (Jiang et al., 2008a). We propose an Integer Linear Programming (ILP) formulation of the word segmentation problem, which naturally provides a word-based model for CWS. Moreover, character-based deterministic constraints, as discussed in Section 4.3.1, can be

easily applied to constrain the solution of this ILP problem.

4.3.4 An ILP formulation of CWS

Given a character sequence $\mathbf{c}$=$c_1...c_n$, there are $s(= n(n+1)/2)$ possible words that are contiguous subsets of $\mathbf{c}$, i.e. $w_1, ..., w_s \subseteq \mathbf{c}$. Our goal is to find an optimal solution

$$\mathbf{x} = x_1...x_s \text{ that maximizes } \sum_{i=1}^{s} score(w_i) \cdot x_i, \tag{4.3.3}$$

subject to

$$(1) \sum_{i:c \in w_i} x_i = 1, \quad \forall c \in \mathbf{c};$$

$$(2) x_i \in \{0, 1\}, \quad 1 \leq i \leq s.$$

The boolean value of x_i, as guaranteed by constraint (2), indicates whether w_i is selected in the segmentation solution or not. Constraint (1) requires every character to be included in exactly one selected word, thus guarantees a proper segmentation of the whole sequence. Take $n = 2$ for example, i.e. $\mathbf{c} = c_1c_2$, the set of possible words is $\{c_1, c_2, c_1c_2\}$, i.e. $s = |\mathbf{x}| = 3$. There are only two possible solutions subject to constraints (1) and (2), $\mathbf{x} = 110$ giving an output set $\{c_1, c_2\}$, or $\mathbf{x} = 001$ giving an output set $\{c_1c_2\}$.

The efficiency of solving this problem depends on the number of possible words (contiguous subsets) over a character sequence, i.e. the number of components of $\mathbf{x}$. So as to reduce the number of components $|\mathbf{x}|$, we use deterministic constraints to predict `IB` tags first. Possible words are then generated with respect to the partially tagged character sequence. More specifically, a character that is tagged with B should always occur at the

beginning of a possible word. Suppose that m characters are tagged with `B` and the maximum distance from one character of tag `B` to the next character of tag `B` is l. The number of possible words for this input sequence is upper bounded by $m(l(l+1)/2)$. For example, for a raw input sequence of characters with length $n = 11$, the number of possible words is 55, i.e. $|\mathbf{x}| = 55$; in contrast, if four of the input characters are tagged with `B` as illustrated in Table 4.8, the number of possible words, i.e. $|\mathbf{x}|$, is reduced from 55 to 18.

n=11	|x|	input
raw	55	法 正 研 究 从 波 黑 撤 军 计 划
constrained	18	法 正 研**/B** 究 从 波**/B** 黑 撤**/B** 军 计**/B** 划

Table 4.8: Comparison of raw input and constrained input.

4.3.5 Experiments on Chinese word segmentation

In this section, we experiment with both character-based and word-based models for the Chinese word segmentation problem. The same dataset with the experiments on character-based deterministic constraints is used, i.e. the one descried in Section 4.3.2. We evaluate word segmentations by F-measure that is calculated as follows:

$$prec = \frac{\#\text{correctly predicted words}}{\#\text{all word predictions}}$$

$$recall = \frac{\#\text{correctly predicted words}}{\#\text{all words}}$$

$$\text{F-measure} = \frac{2 * prec * recall}{prec + recall}.$$

One training schema and two decoders

For tagging Chinese characters, we use the same baseline model as for English POS tagging, i.e. a perceptron-like training schema and the Viterbi algorithm for decoding. The same training schema can also be used to learn word-based CWS models, e.g. the ILP formulation proposed in this work.

More specifically, we use the following linear model for scoring predictions:

$$score(\mathbf{y}) = \theta^T \phi(\mathbf{x}, \mathbf{y}), \tag{4.3.4}$$

where $\phi(\mathbf{x},\mathbf{y})$ is a high-dimensional binary feature representation of $\mathbf{y}$ over input $\mathbf{x}$ and θ contains weights of these features. Parameter θ can be estimated by the averaged perceptron as described in (Collins, 2002). This training algorithm relies on the choice of decoding algorithm. When we experiment with different decoders, e.g. a Viterbi decoder or an ILP solver, the parameter weights in use are trained with the same decoding algorithm.

Like other tagging problems, Viterbi-style decoding is widely used for character-based tagging for CWS. On the other hand, we proposed an ILP formulation of the CWS problem in Section 4.3.4, which naturally provides a word-based CWS model. We can easily tag each character with its relative positions in words, thus the word segmentation output can be deterministically transformed to character-based tagging output with any selected tagset. From this view, the highest scoring tagging sequence can be computed as an ILP problem subject to structural constraints, giving us an inference alternative to Viterbi decoding. For example, take the example of input character sequence $\mathbf{c} = c_1 c_2$. There are 4 tag sequences evaluated by a Viterbi decoder in searching of the highest scoring sequence: `BI`, `BB`, `IB` and

II. In contrast, recall the discussion in Section 4.3.4. There is an extra global constraint implicitly imposed in word-based CWS models, which requires the output tagged sequence to form a legal segmentation. Thus, there are only two tag sequences evaluated by an ILP solver, BB and BI, which correspond to two possible segmentations $\{c_1, c_2\}$ and $\{c_1c_2\}$.

Feature sets

Word segmentation output can be deterministically transformed to character-based tagging sequences, thus character-based CWS features can be directly used in word-based CWS models. Consider a character-based feature function $\phi(c, t, \mathbf{c})$ that maps a character-tag pair to a high-dimensional feature space, with respect to an input character sequence **c**. For a possible word over **c** of length l , $w_i = c_{i_0}...c_{i_0+l-1}$, tag each character c_{i_j} in this word with a character-based tag t_{i_j}. Character-based features of w_i can be computed as $\{\phi(c_{i_j}, t_{i_j}, \mathbf{c})|0 \leq j < l\}$. The first row of Table 4.9 illustrates character-based features of a word of length 3, which is tagged with tagset BMES.

We adopt the 'non-lexical-target' feature templates used in (Jiang et al., 2008a), which generate character-based features for learning. Let c_i denote the i_{th} character relative to the current character c_0 and t_0 denote the tag assigned to c_0. The following templates are used:

$$c_i\&t_0(i = -2...2), \quad c_ic_{i+1}\&t_0(i = -2...1) \quad \text{and } c_{-1}c_1\&t_0.$$

Word-based feature templates usually include the word itself, sub-words contained in the word, contextual characters/words and so on. It has been shown that combining the use of character- and word-based features helps improve performance. However, in the

character-based tagging formulation, word-based features are non-local. To incorporate these non-local features and make the search tractable, various efforts have been made. For example, Jiang et al. (2008a) combine different levels of knowledge in an outside linear model of a two-layer cascaded model; Jiang et al. (2008b) uses the forest re-ranking technique (Huang, 2008); and in (Kruengkrai et al., 2009), only known words in vocabulary are included in the hybrid lattice consisting of both character- and word-level nodes.

When character-based features are incorporated into a word-based model, some word-based features are no longer of interest, such as the starting character of a word, sub-words contained in the word, contextual characters and so on. In this work, we only consider word count as an addition to character-based features, following the idea of using web-scale features in previous work, e.g. (Bansal and Klein, 2011). For a possible word w, let $count(w)$ the times word w occurs in training data. The word count number is further processed following (Bansal and Klein, 2011), $wc(w) = floor(log(count(w)) * 5)/5$. In addition to $wc(w_i)$, we also use corresponding word count features of possible words that are composed of the boundary and contextual characters of w_i. The specific word-based feature templates are illustrated in the second row of Table 4.9.

character-based	$\phi(c_{i_0}, \mathtt{B}, \mathbf{c}), \phi(c_{i_1}, \mathtt{M}, \mathbf{c}), \phi(c_{i_2}, \mathtt{E}, \mathbf{c})$
word-based	$wc(c_{i_0}c_{i_1}c_{i_2})$, $wc(c_lc_{i_0})$, $wc(c_{i_2}c_r)$

Table 4.9: Character-based and word-based features of a possible word w_i . Suppose that $w_i = c_{i_0}c_{i_1}c_{i_2}$, and its preceding and following characters are c_l and c_r respectively.

	precision	recall	F-measure
Viterbi	0.971	0.966	0.968
ILP	0.970	0.977	**0.974**
(Jiang et al., 2008a), POS-			0.971
(Jiang et al., 2008a), POS+			0.973

Table 4.10: F-measure on Chinese word segmentation. Only character-based features are used. POS-/+: perceptron trained without/with POS.

Results

Tagset `BMES` is used for character-based tagging as well as for mapping words to character-based feature space. We use the same Viterbi decoder as implemented for English POS tagging and use a non-commercial ILP solver included in GNU Linear Programming Kit (GLPK), version 4.3. As shown in Table 4.10, when the same feature set is used, the ILP solver returns more accurate segmentations than the Viterbi decoder. More specifically, the F-measure is improved by a relative error reduction of 18.8%, from 0.968 to 0.974.

These results are compared to the core perceptron trained without POS in (Jiang et al., 2008a). They only report results with 'lexical-target' features, a richer feature set than the one we use here. As shown in Table 4.10, we achieve higher performance even with more compact features. Joint inference of CWS and Chinese POS tagging is popularly studied in recent work, e.g. (Ng and Low, 2004), (Jiang et al., 2008a), and (Kruengkrai et al., 2009). It has been shown that better performance can be achieved with joint inference, e.g.

F-measure 0.978 by the cascaded model in (Jiang et al., 2008a). We focus on the task of word segmentation only in this work and show that a comparable F-measure is achievable in a much more efficient manner. Sun (2011) uses the stacked learning technique to merge different levels of predictors, obtaining a combined system that beats individual ones.

Word-based features can be easily incorporated, since the ILP formulation is more naturally viewed as a word-based model. We extend character-based features with the word count features as described in the above section. Currently, we only use word counts computed from training data, i.e. still a closed test. The addition of these features makes a moderate improvement on the F-measure, from 0.974 to 0.975.

As discussed in Section 4.3.4, if we are able to determine that some characters always start new words, the number of possible words is reduced, i.e. the number of variables in an ILP solution is reduced. As shown in Table 4.11, when character sequences are partially tagged by deterministic constraints, the number of possible words per sentence, i.e. avg. $|\mathbf{x}|$, is reduced from 1290.4 to 83.7. This reduction of ILP problem size has a very important impact on the efficiency. As shown in Table 4.11, when taking constrained input, the segmentation speed is increased by **107** times over taking raw input, from 113 characters per second to 12,190 characters per second on a dual-core 3.0HZ CPU.

Deterministic constraints predicting `IB` tags are only used here for constraining possible words. They are very accurate as shown in Section 4.3.2. Few gold predictions are missed from the constrained set of possible words. As shown in Table 4.11, F-measure is not affected by applying these constraints, while the efficiency is significantly improved.

	F-measure	avg. $\|\mathbf{x}\|$	#char per sec.
raw	0.974	1290.4	113 (1×)
constrained	0.974	83.75	**12190** (**107**×)

Table 4.11: ILP problem size and segmentation speed.

4.4 Related work

We are going to review related work in three categories for this chapter: previous work on English POS tagging, previous work on Chinese word segmentation and previous work on deterministic algorithms.

POS tagging. POS tagging, especially for English, is a very well studied problem and a English POS tagger with an accuracy above 97% can be easily built in a couple of hours. More important, this problem has been widely studied for the introduction of new statistical models of sequence classification problems, the introduction of new learning frameworks and the introduction of new searching strategies. For example, influent models such as Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Lafferty et al., 2001) are usually introduced to NLPers with experiments on POS tagging; Collins (2002) proposed a very generally used, perceptron like learning framework with POS tagging as a main application; I personally get acquainted with the Maximum Entropy framework (Ratnaparkhi, 1996) and Support Vector Machines (SVM) (Joachims, 2008) through experiments on POS tagging; and search strategies such as bidirectional (Shen et al., 2007) and easiest-first (Tsuruoka and Tsujii, 2005) are all first studied by the authors in experi-

ments on POS tagging and then other applications in their following work (Shen and Joshi, 2008; Tsuruoka et al., 2011). Thus, when we are to explore the linguistic concept of determinism as deterministic constraints in NLP applications, the first application came up in our mind is POS tagging.

Chinese word segmentation. We then explore deterministic constraints for a more challenging task, Chinese word segmentation. In the earliest work on Chinese word segmentation, word-based CWS models are more popular, e.g. words are found one by one according to forward maximum match (Sproat et al., 1996). Since Xue (2003) proposed to tag each character with relative position information, character-based CWS models become dominant and all popular techniques proposed for sequence classification problem are naturally applicable to Chinese word segmentation, such as Maximum Entropy approaches (Low et al., 2005), CRF (Tseng et al., 2005) and so on. Once classified as yet another sequence classification problem, CWS may lose attention, however, recent research on Chinese word segmentation are very active, and especially on the joint inference of word segmentation and POS tagging, e.g. (Jiang et al., 2008a; Kruengkrai et al., 2009; Zeng et al., 2013). In recent work, word-based models also retrieve certain interests, e.g. (Sun, 2011; Zhang and Clark, 2007; Ng and Low, 2004). It is to our surprise that, the ILP formulation of this word segmentation problem has never been tried before, which easily achieves the state-of-art performance above 97% as we have shown in this work.

Deterministic algorithms. At least up to our knowledge of syntax-related NLP applications such as POS tagging and (constituent/dependency) parsing, when deterministic algorithms are considered for probabilistic inference, it always refers to a greedy search, i.e. a beam search with the beam of one. Since POS taggers already run fast with modern processors, deterministic search seems mainly interesting for parsing, since speedup for parsing is still of practical interest (Wang et al., 2006). Furthermore, as argued in (Nivre and R.McDonald, 2008), with a greedy search, richer features are more easily employed, thus to some degree compensate the loss of performance due to the approximate search. As we have already discussed, the deterministic constraints proposed here manifest determinism in language, so they differ with and can work well with deterministic search which is only a statistical pruning technique. The most similar idea with these deterministic constraints, as we found in syntax-related applications, is the hard constraints of chart parsing proposed by Roark and Hollingshead (2009). They have also noticed that, in a pipeline system where the following search is constrained by preprocessed constraints, the high precision of these constraints is crucial to the overall performance. However, they tolerate much less accurate constraints than us to constrain the following probabilistic search, so the concept of determinism plays no role in their work. The use of hard constraints is shown to improve both performance and efficiency in their experiments on parsing, but only compared to a very fundamental baseline model. In contrast, we have applied the deterministic constraints to the state-of-art POS tagging and Chinese word segmentation models, and also achieved improvements in both performance and efficiency. In the sense of capturing determinism in language, this work is mainly motivated by early work on deterministic

parsing, e.g. Marcus (1980)'s parser which deterministically interprets most context-free grammars with the help of lookahead features. It may be interesting to note that, NLPers have been exercising with more and more advanced techniques to deal with ambiguity and nondeterminism in natural languages, however, linguists are still making effort on inventing new representations of linguistic structures to remove nondeterminism theoretically, e.g. Chomsky (2007)'s Phase theory.

4.5 Conclusion and future work

We have shown by experiments that large number of deterministic constraints can be learned from training examples, as long as the proper representation is used. These deterministic constraints are very useful in constraining probabilistic search, for example, they may be directly used for determining predictions as in English POS tagging, or used for reducing the number of variables in an ILP solution as in Chinese word segmentation. The most notable advantage in using these constraints is the increased efficiency. The two applications are both well-studied; there isn't much space for improving accuracy. Even so, we have shown that as tested with the same feature set for CWS, the proposed ILP formulation significantly improves the F-measure as compared to Viterbi decoding.

These two simple applications suggest that it is of interest to explore data-driven deterministic constraints learnt from training examples. There are more interesting ways in applying these constraints, which we are going to study in future work.

Chapter 5

Conclusion and personal reflections

We have explored the distinction between closed- and open-class words in Chapter 2, long-tail word distributions in Chapter 3, and deterministic constraints in Chapter 4. In this chapter, we will first summarize the engineering achievements we obtain through the exploration of these linguistic aspects and then discuss our lessons in this line of research.

5.1 Engineering achievements

Along our exploration of some linguistic aspects, we achieve the following improvements for certain NLP applications.

- The distinction between closed-class and open-class words is crucial in practicing a feature-based view of POS tags, which suggests a new formulation of POS tagging that requires less resource to learn. As an application of this distinction, we proposed a totally unsupervised POS tagging system which achieves comparable performance

with those unsupervised POS tagging systems that require the input of a dictionary.

- Even though we are not the first to explore long-tail distributions for morphology learning, instead of explaining off this distribution, our proposed model is the first that can actually take advantage of the real word distribution. By using the log-normal distribution to model the long-tail distributions, we achieve the state-of-the-art performance for a token-based evaluation of the English verb inflections.

- In contrast to pruning, deterministic constraints on probabilistic inference speed up searching without a trade-off in performance. Instead of composing deterministic constraints with expert knowledge, we propose to learn deterministic constraints in a data-driven way. For POS tagging, the learnt deterministic constraints resolve more than 80% of the tagging predictions. These predictions can not only reduce the search space for the following decoding process but also provide additional features for the following statistical inference. For the problem of Chinese word segmentation, the learnt character-based constraints are used to reduce the search space of a word-based CWS model. So as to use these constraints in a natural way, we propose an ILP formulation of the word segmentation problem. While matching the state-of-the-art performance for both applications, the proposed use of deterministic constraints speeds up the Viterbi decoder for English POS tagging by a factor of 10 and speeds up the ILP solver for Chinese word segmentation by a factor of 100.

As implied in the above summarization, for supervised NLP tasks, the heavily lexicalized models that learn well from large annotated corpora have achieved notable success in

fundamental applications such as POS tagging and word segmentations. In other words, it becomes harder and harder to achieve new state-of-the-art performance for these applications. However, the same problem formulation or selected model that work well for a supervised learning task may not be appropriate for unsupervised learning of the same problem. Thus we need to resort to the linguistic aspects of these NLP applications and consider new approaches to attack these problems.

One engineering difficulty in attacking these unsupervised learning problems is the chaos in evaluation. For supervised learning tasks, we have become used to using the standard data resource, following the standard annotation guidelines and playing the standard games. However, for unsupervised learning, it is hard to produce results in a 'standard' form to compare, since we are now learning from raw text only but not standardized data. We argue that, the best way to evaluate the product of a unsupervised learning is to use it in a more advanced application. For example, we evaluate the acquisition of closed-class words in unsupervised POS tagging. And we suggest that our (totally) unsupervised POS tagging of the core lexical categories makes more sense for unsupervised learning of more complex linguistic structures. Moreover, our work on morphology learning pays attention to token-based input and token-based evaluations, but there is no token-based data resource with morphological annotations for English. Thus, following (Goldwater et al., 2006), we build gold standards for English inflections from the POS annotation of the Penn Treebank. This makes our work difficult to compare since most work on morphology learning competes for type-based evaluations only. It took us a long time to fully understand why our unsupervised work was most appreciated by reviewers who share the same strong interest

in linguistics. And it gradually emerges as a big picture to finish so as to achieve something not affected by the chaos in evaluation for unsupervised work.

On the other hand, we have also explored linguistic expects for supervised tasks, and our main contribution lies in the improvement of efficiency. And more importantly, the proposed deterministic constraints improve efficiency without a trade-off in performance. Even though in applications such as POS tagging or word segmentation, efficiency does not bother researchers too much, it is certainly worth further exploration of these deterministic constraints for more advanced applications such as parsing and machine translation. Moreover, our experiments have led us to consider less popular machine learning frameworks such as Integer Linear Programming, which has many advantages but not in efficiency.

5.2 Personal reflections on linguistics and engineering

Even though not included in this dissertation, we have a formal syntax study (Zhao, 2010) of a special functional word in Chinese, BA, which, simply speaking, marks causatives in Chinese. This work has two important effects on my following work: first, I realized that I am an engineer and to speak as a linguist is hard; second, it intrigued my interest in functional words and determinism in language. Then, so as to explore functional words and determinism in language, I choose unsupervised POS tagging and word segmentation as the playground. It is not a traditional way to build one's research line, and I won't recommend it. More typically, in our field, one's dissertation concentrates either on a specific application or on a specific technique. After reviewing classic work in our field, I

realize that most breakthroughs are actually motivated by proper linguistic considerations, so our path does not have any exclusive advantage in getting the linguistic sense.

Then I wonder for a long time whether my training in linguistics helps or not for my NLP work. It is a real question because a possible answer is that my linguistic background may actually constrain my steps. For example, if I was not aware that "Zipf's law" is almost interchangeably used with the term 'long-tail distribution' in vocabulary study, I won't devote so much effort to examine whether there is any theoretical aspect favoring Zipf's law over log-normal distributions in vocabulary study. Only after this study, we feel free to propose a new model for long-tail distributions, but for a typical engineering study, as long as performance is improved, such an exploration is not necessary and to try a new distribution is a piece of cake for engineers. However, without a formal training in syntax, I won't have enough background to challenge the most popular formulation of POS tagging. Our feature-based view of POS tags not only suggests a new framework for unsupervised POS tagging but also suggests the patterns to learn deterministic constraints for supervised POS tagging. As shown by our experiments in Chinese word segmentation, proper deterministic constraints are not so easy and natural to learn, and that our first try works in English POS tagging is not luck but a result of our previous work.

Another doubt of my obsession in linguistics comes from its bad effect in my writing. When presenting my early work to the NLP community, I unintentionally assumed my audience has sympathy for linguistics. Later, I note that it is no longer popular to express one's linguistic opinion, instead, linguistic motivations are expressed as formal characteristics of data. From this point of view, natural language is simply another kind of data, so

that the experiments on NLP applications with advanced machine learning techniques lead to more general conclusions. It has been a very effective track, but the problem is that, compared to the amazing improvements our field achieved when machine learning techniques were first introduced, recent achievements on this track are less and less exciting. In this context, we may conclude this dissertation by justifying our twisted line of research from the following angle: explorations (on linguistic aspects) ought to be launched, one of them may discover the new continent (even though most of them will fail).

Appendix A

Examples of the deterministic constraints

In this appendix, we provide more details of the deterministic constraints learnt with bigram and trigram patterns. In Figure A.1, we show the top 10 constraints that actually resolve ambiguity for English POS tagging. When tagging the 56684 words in the test data, the average number of deterministic constraints fired for each word is 7.3, e.g. for 6664 words there are 16 deterministic constraints applied to their contexts. For the 48276 words occurring in the contexts that evoke deterministic rules, only 68 of them are not tagged at the first stage due to the inconsistency between deterministic rules. On the other hand, since there is no morphological feature composed in the patterns that generate deterministic constraints for Chinese word segmentation, the average number of deterministic constraints fired for each character is only 1.36. When tagging Chinese characters with the IB tagset, there are at most 3 constraints fired for the same word occurrences, and there are 2502 such

occurrences. Similar with English POS tagging, there are only 13 word occurrences where the fired deterministic constraints do not agree on the same prediction. In Figure A.2, we show the top 10 constraints that predict the beginning of a word.

```
DT|NNP|VB@0+.@-1 ==> DT [615]
the@0+IN|RB|RP@-1 ==> DT [592]
IN|RB|RP@0+the@1 ==> IN [592]
IN|RB|RP@0+DT|JJ|NN|NNP|VBP@1 ==> IN [592]
DT|JJ|NN|NNP|VBP@0+IN|RB|RP@-1 ==> DT [592]
the@0+NN@1 ==> DT [575]
DT|JJ|NN|NNP|VBP@0+NN@1 ==> DT [575]
IN|RB|RP@0+NN@-1 ==> IN [514]
the@0+,@-1 ==> DT [428]
DT|JJ|NN|NNP|VBP@0+,@-1 ==> DT [428]
```

Figure A.1: Examples of deterministic constraints for English POS tagging, and the count of how many times each rule is fired when tagging the test data is given in [].

```
。@0+#1# ==> B [252]
发@0+展@1 ==> B [57]
，@0+说@-1 ==> B [43]
）@0+#1# ==> B [43]
中@0+，@-1 ==> B [38]
经@0+济@1 ==> B [38]
，@0+中@1 ==> B [38]
合@0+作@1 ==> B [34]
问@0+题@1 ==> B [32]
电@0+日@-1 ==> B [32]
```

Figure A.2: Examples of deterministic constraints for Chinese word segmentation, and the count of how many times each rule is fired when tagging the test data is given in [].

Bibliography

Abend, O., Reichart, R., and Rappoport, A. (2010). Improved unsupervised POS induction through prototype discovery. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Aldous, D. J. (1985). Exchangeability and related topics. *Lecture Notes in Mathematics*, 1117:1–198.

Banko, M. and Moore, R. C. (2004). Part-of-speech tagging in context. In *Proceedings of the 20th international conference on Computational Linguistics*, page 556. Association for Computational Linguistics.

Bansal, M. and Klein, D. (2011). Web-scale features for full-scale parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 693–702.

Bates, E. and Wulfeck, B. (1989). Crosslinguistic studies of aphasia. *The crosslinguistic study of sentence processing*, pages 328–371.

Biassou, N., Obler, L. K., Nespoulous, J.-L., Dordain, M., and Harris, K. S. (1997). Dual processing of open- and closed-class words. *Brain and Language*, 57(3):360–373.

Blunsom, P. and Cohn, T. (2011). A hierarchical Pitman-Yor process HMM for unsupervised part-of-speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 865–874. Association for Computational Linguistics.

Bradley, D. C. (1978). *Computational distinctions of vocabulary type.* PhD thesis, Massachusetts Institute of Technology.

Chahuneau, V., Smith, N. A., and Dyer, C. (2013). Knowledge-rich morphological priors for Bayesian language models. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1206–1215, Atlanta, Georgia. Association for Computational Linguistics.

Chan, E. (2008). *Structures and distributions in morphology learning*. PhD thesis, University of Pennsylvania.

Chomsky, N. (1970). Remarks on nominalization. In Jacobs, R. and Rosenbaum, P., editors, *Readings in English Transformational Grammar*, pages 184–221. Ginn.

Chomsky, N. (2007). Approaching UG from below. *Interfaces + recursion = language*, pages 1–29.

Christodoulopoulos, C., Goldwater, S., and Steedman, M. (2010). Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 conference on Empirical methods in natural language processing(EMNLP)*.

Chung, C. and Pennebaker, J. W. (2007). The psychological functions of function words. *Social communication*, pages 343–359.

Clark, A. (2003). Combining distributional and morphological information for part-of-speech induction. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 59–66. Association for Computational Linguistics.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of conference on Empirical methods in natural language processing(EMNLP)*, pages 1–8.

Creutz, M. (2003). Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 280–287. Association for Computational Linguistics.

Creutz, M. and Lagus, K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.

Dasgupta, S. and Ng, V. (2007). High-performance, language-independent morphological segmentation. In *HLT-NAACL*, pages 155–163.

Dempster, A. P., Laird, N. M., Rubin, D. B., et al. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal statistical Society*, 39(1):1–38.

Friederici, A. D., Pfeifer, E., and Hahne, A. (1993). Event-related brain potentials during natural speech processing: Effects of semantic, morphological and syntactic violations. *Cognitive brain research*, 1(3):183–192.

Friederici, A. D., Rüschemeyer, S.-A., Hahne, A., and Fiebach, C. J. (2003). The role of left inferior frontal and superior temporal cortex in sentence comprehension: localizing syntactic and semantic processes. *Cerebral cortex*, 13(2):170–177.

Friederici, A. D. and Schoenle, P. W. (1980). Computational dissociation of two vocabulary types: Evidence from aphasia. *Neuropsychologia*, 18(1):11–20.

Garrette, D. and Baldridge, J. (2013). Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of NAACL-HLT*, pages 138–147.

Goldberg, Y., Adler, M., and Elhadad, M. (2008). EM can find pretty good HMM POS-taggers (when given a goodstart). In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Goldsmith, J. (2001). Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

Goldwater, S. and Griffiths, T. (2007). A fully Bayesian approach to unsupervised part-of-speech tagging. In *Annual meeting-association for computational linguistics*, volume 45, page 744.

Goldwater, S., Griffiths, T., and Johnson, M. (2006). Interpolating between types and tokens by estimating power-law generators. In *NIPS*.

Goodman, J. C., Dale, P. S., and Li, P. (2008). Does frequency count? Parental input and the acquisition of vocabulary. *Journal of child language*, 35(3):515.

Gordon, B. and Caramazza, A. (1982). Lexical decision for open- and closed-class words: Failure to replicate differential frequency sensitivity. *Brain and Language*, 15(1):143–160.

Graca, J., Ganchev, K., Taskar, B., and Pereira, F. (2009). Posterior vs. parameter sparsity in latent variable models. In *Advances in Neural Information Processing Systems 22 (NIPS)*.

Graça, J. (2011). Language-independent learning of open and closed word categories. private communication.

Haghighi, A. and Klein, D. (2006). Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 320–327. Association for Computational Linguistics.

Huang, L. (2008). Forest reranking: Discriminative parsing with non-local features. In *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

Ishwaran, H. and James, L. F. (2003). Generalized weighted Chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13:1211–1235.

Jiang, W., Huang, L., Liu, Q., and Lü, Y. (2008a). A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

Jiang, W., Mi, H., and Liu, Q. (2008b). Word lattice reranking for Chinese word segmentation and part-of-speech tagging. In *Proceedings of the International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 385–392.

Joachims, T. (2008). Sequence tagging with structural support vector machines. `http://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html`.

Jordan, M. I. (2010). The conjugate prior for the Normal distribution. `http://www.cs.berkeley.edu/~jordan/courses/260-spring10/lectures/lecture5.pdf`. Lecture notes on Stat260: Bayesian Modeling and Inference.

Joshi, A. K. (1982). Processing of sentences with intra-sentential code-switching. In *Proceedings of the 9th conference on Computational linguistics-Volume 1*, pages 145–150. Academia Praha.

Kristjansson, T., Culotta, A., and Viola, P. (2004). Interactive information extraction with constrained conditional random fields. In *In AAAI*, pages 412–418.

Kruengkrai, C., Uchimoto, K., Kazama, J., Wang, Y., Torisawa, K., and Isahara, H. (2009). An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL '09, pages 513–521.

Lada, A. (2002). Zipf, power-laws, and Pareto - a ranking tutorial. `http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html`.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, San Fransisco. Morgan Kaufmann.

Lamar, M., Maron, Y., and Bienenstock, E. (2010). Latent-descriptor clustering for unsupervised POS induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 799–809. Association for Computational Linguistics.

Lee, Y. K., Haghighi, A., and Barzilay, R. (2011). Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computa-*

tional Natural Language Learning, Portland, Oregon, USA. Association for Computational Linguistics.

Lignos, C. (2010). Learning from unseen data. In *Proceedings of Morpho Challenge 2010*, pages 35–38, Helsinki, Finland.

Lignos, C., Chan, E., Marcus, M. P., and Yang, C. (2009). A rule-based unsupervised morphology learning framework. In *Working Notes for the CLEF 2009 Workshop*.

Low, J. K., Ng, H. T., and Guo, W. (2005). A maximum entropy approach to Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 1612164.

Luhn, H. P. (1958). *Auto-encoding of documents for information retrieval systems*. IBM Research Center.

Maffi, L. (2007). Implementation of K-means clustering in Python. `http://www.fantascienza.net/leonardo/so/kmeans/kmeans.html`.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Marcus, M. P. (1980). *Theory of syntactic recognition for natural languages*. MIT press.

Martins, A. F. T., Smith, N. A., and Xing, E. P. (2009). Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the*

47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP), pages 342–350, Singapore.

Miller, G. A. (1957). Some effects of intermittent silence. *American Journal of Psychology*, 70:311–314.

Miltsakaki, E., Prasad, R., Joshi, A. K., and Webber, B. L. (2004). The Penn Discourse Treebank. In *LREC*.

Mitzenmacher, M. (2004). A brief history of generative models for power law and lognormal distributions. *INTERNET MATHEMATICS*, 1:226–251.

Moon, T., Erk, K., and Baldridge, J. (2009). Unsupervised morphological segmentation and clustering with document boundaries. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 668–677, Stroudsburg, PA, USA. Association for Computational Linguistics.

Moon, T., K.Erk, and Baldridge, J. (2010). Crouching Dirichlet, hidden Markov model: unsupervised POS tagging with context local tag generation. In *In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Neville, H. J., Mills, D. L., and Lawson, D. S. (1992). Fractionating language: Different neural subsystems with different sensitive periods. *Cerebral Cortex*, 2(3):244–258.

Newman, M. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5):323–351.

Ng, H. T. and Low, J. K. (2004). Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 277C284.

Nivre, J. and R.McDonald (2008). Integrating graph-based and transition-based dependency parsers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 950–958.

Perline, R. (1996). Zipfs law, the central limit theorem, and the random division of the unit interval. *Physical Review*, 54(1):220–223.

Pitman, J. and Yor, M. (1997). The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900.

Poon, H., Cherry, C., and Toutanova, K. (2009). Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217. Association for Computational Linguistics.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *In Proceedings of the Empirical Methods in Natural Language Processing Conference (EMNLP)*.

Ravi, S. and Knight, K. (2009). Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Reichart, R., Fattal, R., and A.Rappoport (2010). Improved unsupervised POS induction using intrinsic clustering quality and a zipfian constraint. In *CoNLL*.

Roark, B. and Hollingshead, K. (2009). Linear complexity context-free parsing pipelines via chart constraints. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 647–655. Association for Computational Linguistics.

Roth, D. and Yih, W. (2005). Integer linear programming inference for conditional random fields. In *In Proceedings of the International Conference on Machine Learning (ICML)*, pages 737–744.

Schütze, H. (1993). Part-of-speech induction from scratch. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 251–258. Association for Computational Linguistics.

Shen, L. and Joshi, A. K. (2008). Ltag dependency parsing with bidirectional incremental construction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing(EMNLP)*, pages 495–504. Association for Computational Linguistics.

Shen, L., Satta, G., and Joshi, A. K. (2007). Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Simon, H. A. (1955). On a class of skew distribution functions. *Biometrika*, 42(3/4):425–440.

Smith, N. A. and Eisner, J. (2005). Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics.

Sproat, R., Gale, W., Shih, C., and Chang, N. (1996). A stochastic finite-state word-segmentation algorithm for Chinese. *Comput. Linguist.*, 22(3):377–404.

Sun, W. (2011). A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of the ACL-HLT 2011*.

Teichert, A. and Daume, H. (2010). Unsupervised part-of-speech tagging without a lexicon. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*.

Toutanova, K. and Johnson, M. (2008). A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *NIPS*, pages 1521–1528.

Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 171. Jeju Island, Korea.

Tsuruoka, Y., Miyao, Y., and Kazama, J. (2011). Learning with lookahead: Can history-based models rival globally optimized models? In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 238–246, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tsuruoka, Y. and Tsujii, J. (2005). Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 467–474. Association for Computational Linguistics.

Wang, M., Sagae, K., and Mitamura, T. (2006). A fast, accurate deterministic parser for Chinese. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 425–432. Association for Computational Linguistics.

Wang, Y., Xiang, J., Kotecha, R., Vannest, J., Liu, Y., Rose, D., Schapiro, M., and Degrauw, T. (2008). Spatial and frequency differences of neuromagnetic activities between the perception of open- and closed-class words. *Brain topography*, 21(2):75–85.

Xue, N. (2003). Chinese word segmentation as character tagging. *International Journal of Computational Linguistics and Chinese Language Processing*, 9(1):29–48.

Zeng, X., Wong, D. F., Chao, L. S., and Trancoso, I. (2013). Graph-based semi-supervised model for joint Chinese word segmentation and part-of-speech tagging. In *Proceeding of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 770–779.

Zhang, Y. and Clark, S. (2007). Chinese Segmentation with a Word-Based Perceptron Algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847.

Zhao, Q. (2010). Ba as spell-out of little v. In *The 6th International Workshop on Theoretical East Asian Linguistics (TEAL-6)*.

Zhao, Q. and Marcus, M. (2009). A simple unsupervised learner for POS disambiguation rules given only a minimal lexicon. In *Proceedings of the 2009 conference on Empirical methods in natural language processing(EMNLP)*.

Zhao, Q. and Marcus, M. (2011). Functional elements and POS categories. In *IJCNLP*, pages 1198–1206.

Zhao, Q. and Marcus, M. (2012a). Exploring deterministic constraints: From a constrained English POS tagger to an efficient ILP solution to Chinese word segmentation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 1054–1062, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhao, Q. and Marcus, M. (2012b). Long-tail distributions and unsupervised learning of morphology. In *Proceedings of the International Conference on Computational Linguistics*, pages 3121–3136.

Zipf, G. K. (1949). *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, Reading, MA.